J. Elsing H. Sterner A. Wagner

GRAFIK AUF DEM COMMODORE 64

Anregungen und Erläuterungen in BASIC



J. Elsing H. Sterner A. Wagner

GRAFIK AUF DEM COMMODORE 64

Anregungen und Erläuterungen in BASIC



CIP-Kurztitelaufnahme der Deutschen Bibliothek

Elsing, Jürgen:
Grafik auf dem Commodore 64: Anregungen u. Erl.
in BASIC/J. Elsing; H. Sterner; A. Wagner. –
Vaterstetten: IWT, 1983,
ISBN 3-88322-027-2

NE: Sterner, Heinz:; Wagner, Annette:

ISBN 3-88322-027-2
1. Auflage 1983

Autor: J. Elsing; H. Sterner; A. Wagner
Umschlag: Kaselow + Partner
Herstellung: Freiburger Graphische Betriebe, Freiburg

V o r w o r t

Heutzutage bieten bereits viele Micro-Computer die Möglichkeit zu grafischen Darstellungen. In dieser Hinsicht zeichnet sich insbesondere der Commodore 64 aus, der eine ganze Reihe verschiedenartiger Grafik-Darstellungen ermöglicht, die sowohl für berufliche, als auch für Hobby-Anwendungen genutzt werden können.

In diesem Buch sollen die grundlegenden Verfahren zur Erstellung von Grafiken auf dem Commodore 64 beschrieben werden. Dazu sind lediglich einige BASIC-Kenntnisse erforderlich. Leider sind speziell die Grafik-Fähigkeiten des Commodore 64 in dem mitgelieferten Handbuch nicht so ausführlich beschrieben, wie es wünschenswert wäre (dies gilt besonders für die hochauflösende Grafik, die im Commodore-Handbuch überhaupt nicht beschrieben ist.).

Das vorliegende Buch will nun in erster Linie dem "Computer-Neuling" oder "Einsteiger" Anregungen und Hinweise vermitteln, die es ihm ermöglichen, sich die vielfältigen grafischen Möglichkeiten des Commodore 64 zu erschließen.

Die abgedruckten Programm-Beispiele wurden zwar möglichst kurz gehalten, trotzdem ist das Eintippen von Programmen nicht gerade die interessanteste Seite der Beschäftigung mit einem Computer. Deshalb sind die Programme aus diesem Buch (zum Teil zusätzlich in einer erweiterten Version) auch auf Diskette oder Kassette erhältlich.

Für alle Hinweise, Tips, kritische (aber natürlich auch zustimmende) Reaktionen, die einer Verbesserung dieses Buches dienen können, sind wir dankbar!

Jetzt aber wollen wir Sie nicht länger mit der Vorrede aufhalten und wünschen Ihnen viel Spaß und Erfolg bei der Erforschung der grafischen Fähigkeiten Ihres Computers!

Die Autoren

I n h a l t

E i n l e i t u n g

Ausgehend von den einfachen Grafiken mit den 'fest eingebauten' Grafik-Zeichen des Commodore 64 führt das vorliegende Buch systematisch zu den anspruchsvolleren grafischen Gestaltungsmöglichkeiten dieses Computers, illustriert jeweils durch typische Beispiel-Programme.

Die einfachste Möglichkeit, grafische Darstellungen auf den Bildschirm zu bringen, bieten die 'fest eingebauten' Grafik-Zeichen des Commodore 64, die ähnlich wie z.B. Buchstaben oder Ziffern benutzt werden können. Im ersten Kapitel wird anhand einiger kleiner Beispiele (die sich zu einem einzigen Programm zusammensetzen lassen) ausführlich auf diese Grafik-Form eingegangen, mit der sogar einfache Bewegungen dargestellt werden können.

Eine Besonderheit des Commodore 64 stellen die sogenannten 'Sprites' (sprite = 'Kobold') dar. Dabei handelt es sich um kleine Figuren, die der Benutzer selbst definieren kann. Ein Sprite kann leicht an eine beliebige Bildschirmposition gebracht werden bzw. sehr schnell über den Bildschirm bewegt werden. Das zweite Kapitel gibt genaue Anleitungen zum Erstellen dieser Grafik-Figuren.

Im dritten Kapitel wird die hochauflösende Grafik beschrieben. Damit ist es möglich, grafische Darstellungen mit einer Auflösung von 320 x 200 Punkten auf den Bildschirm zu bringen.

Damit Sie die Funktionsweise Ihres Computers besser verstehen können, folgt im vierten Kapitel etwas Theorie.

Das fünfte Kapitel enthält einige ausführlichere Beispiel-Programme, in denen jedoch weitgehend auf die bereits bekannten Programm-'Bausteine' aus den vorigen Kapiteln zurückgegriffen wird.

Anschließend finden Sie noch eine kurze Erläuterung der verwendeten BASIC-Anweisungen sowie verschiedene Tabellen, die Ihnen (ebenso wie ein Stichwortverzeichnis) auch später noch nützlich sein können, wenn Sie Informationen benötigen, die die Grafik-Speicherorganisation des Commodore 64 betreffen.

1. Einfache Grafiken mit den 'fest eingebauten' Grafik-Zeichen des Commodore 64

Die erste Möglichkeit, Grafiken zu erstellen, bieten die fest eingebauten Commodore-Grafik-Zeichen, die Sie als Zweit-Funktionen auf den Tasten sehen. Die Zeichen rechts unten erhält man durch gleichzeitiges Drücken der SHIFT-Taste sowie der Taste, auf der sich das gewünschte Zeichen befindet. Um das Symbol links unten zu erhalten, muß statt der SHIFT- die Commodore-Taste gedrückt werden (neben der linken SHIFT-Taste).

Außerdem läßt sich jedes Zeichen auch noch invers darstellen, d.h., aus einem hellen Zeichen mit dunklem Hintergrund wird ein dunkles Zeichen mit hellem Hintergrund. Dies erreicht man durch gleichzeitiges Drücken der CTRL-Taste und der 9-Taste (= RVS ON). Die Rückkehr zur normalen Darstellung geschieht entsprechend mit CTRL-O (= RVS OFF).

1.1. Erstellen einer Grafik durch PRINT-Befehle. Beispiel-Programm zur direkten Verwendung der Grafik-Tasten.

Mit den 'eingebauten' Grafik-Zeichen lassen sich schon einfache Figuren auf dem Bildschirm zusammensetzen. Dies kann entweder direkt geschehen, indem man den Cursor an die entsprechende Bildschirmposition steuert und dort das gewünschte Zeichen setzt. Die so erstellten Grafiken können aber nicht "aufbewahrt" werden. Um ein Bild auch reproduzieren zu können, muß es durch ein BASIC-Programm aufgebaut werden. Auf diese Weise läßt es sich abspeichern und jederzeit wieder aufrufen. Für die Programmierung solcher Bilder gibt es nun zwei Möglichkeiten:

Einerseits kann man natürlich - genauso wie man beliebige andere Zeichen (wie Buchstaben oder Zahlen) ausgeben kann - auch die "fest eingebauten" Grafik-Zeichen durch PRINT-Befehle auf den Bildschirm bringen. Dabei muß allerdings die Reihenfolge berücksichtigt werden, in der die Zeichen ausgegeben werden sollen.

PRINT-Befehle können nur von oben nach unten und innerhalb einer Zeile von links nach rechts ausgeführt werden. Es ist also mit der PRINT-Anweisung nicht

möglich, zuerst eine Zeichenfolge in Zeile 16 auszugeben und danach ein Wort in Zeile 12 zu schreiben. Außerdem müssen etwa vorkommende Leerstellen ebenfalls berücksichtigt werden. Diese Überlegungen werden im 1. Beispiel verdeutlicht:

```
10 REM.......................................BEISPIEL 1.1
20 PRINT"[CLR]"
25 REM.......................................LEERZEICHEN
30 FOR I=1TO8:PRINT:NEXTI
560 REM DIE VERWENDETEN TASTEN STEHEN IN KLAMMERN HINTER DER ZEILE
580 REM
600 PRINT TAB(32);"_":REM                 (P)
620 PRINT TAB(31);"/\    /|":REM          (N,J,M,U,K,M)
640 PRINT TAB(30);"/ \/ \ //":REM         (N,J,K,M)
660 PRINT TAB(29);"/ \/ \/ \|‾‾|":REM     (N,J,K,O,P)
680 PRINT TAB(28);"/ \/ \/ \/|  |":REM    (N,J,K,H,N)
700 PRINT TAB(27);"/ \/ \/ \/ \/ \":REM   (N,J,K,M)
720 PRINT TAB(26);"/ \/ \/ \/ \/ \/ \":REM (N,J,K,M)
740 PRINT TAB(26);"|‾‾‾‾‾‾‾‾‾‾‾‾|":REM    (O,Y,P)
760 PRINT TAB(26);"|            |":REM    (H,N)
780 PRINT TAB(26);"| ‾‾  ‾‾  ‾‾ |":REM    (H,U,*,I,A,R,S,N)
800 PRINT TAB(26);"| | | ++  ++ |":REM    (H,-,O,+,W,N)
820 PRINT TAB(26);"| |/| __  __ |":REM    (H,-,K,Z,E,X,N)
840 PRINT TAB(26);"‾‾‾‾‾‾‾‾‾‾‾‾‾‾":REM    (Y)
```

Mit etwas Phantasie ist leicht zu erkennen, daß hier ein Haus gezeichnet wird:

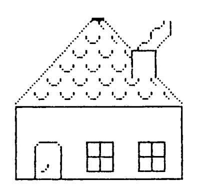


Abb. 1.1 zu Beispiel 1.1

Das geschieht im einzelnen so:

Zeile 20 löscht den Bildschirm. (Das invers dargestellte Herz zwischen den Anführungszeichen entsteht durch Drücken der Tasten: " (Anführungszeichen), SHIFT, CLR/HOME, ").

In Zeile 30 werden 8 Leerzeilen gedruckt.

Die TAB-Anweisungen in den folgenden Zeilen sorgen für die notwendigen Leerzeichen am Anfang einer Reihe.

1.2. Grafik mit PRINT-Befehlen in Verbindung mit der CHR$- und der ASC-Funktion.

Statt die Grafikzeichen eingeschlossen in Anführungszeichen mit einem PRINT-Befehl auszugeben, können die Zeichen auch codiert werden. Dazu lassen sich die CHR$-Werte aus der Tabelle ab S. 122 benutzen. (CHR ist die Abkürzung für character - dt.: Zeichen - und "$" weist darauf hin, daß es sich hierbei um eine String-Funktion (string = Zeichenkette) handelt, d.h. um eine Funktion, die nicht nur auf Zahlen, sondern auf beliebige Zeichen angewandt werden kann). Im Computer sind sämtliche Zeichen mit Hilfe eines Zahlen-Codes verschlüsselt. Das einer bestimmten Zahl entsprechende Zeichen - z.B. Buchstabe oder Ziffer - kann man nun durch die BASIC-Anweisung CHR$ erhalten (wobei in Klammern die entsprechende Zahl anzugeben ist). So ergibt z.B.

```
PRINT CHR$(77)  den Buchstaben 'M',
PRINT CHR$(113) ergibt einen Kreis und
PRINT CHR$(14)  schaltet um auf Kleinbuchstaben.
```

Die Umkehr-Funktion zu CHR$ heißt ASC. Das Argument von ASC (d.h. der in Klammern anzugebende Ausdruck) ist einfach ein Zeichen, das in Anführungszeichen stehen muß. Z.B. liefert die Anweisung

```
PRINT ASC("M") die zugehörige Code-Zahl 77,
PRINT ASC("A") ergibt 65 usw.
```

(ASC steht übrigens für "ASCII-Code", und das wiederum ist die Abkürzung für "American Standard Code for Information Interchange", den die meisten Computer "verstehen").

Mit Hilfe der CHR$-Funktion läßt sich ein weiteres Beispiel jetzt folgendermaßen darstellen:

```
10 REM..................................BEISPIEL 1.2
20 PRINT"□";:REM.............JETZT FOLGEN DIE 'CODIERTEN' ZEICHEN
280 PRINTCHR$(109);CHR$(32);CHR$(109);CHR$(125);CHR$(110);
300 PRINTCHR$(32);CHR$(110):PRINTCHR$(32);
320 PRINTCHR$(109);CHR$(110);CHR$(163);CHR$(109);CHR$(110)
340 PRINTCHR$(96);CHR$(96);CHR$(116);CHR$(32);CHR$(167);CHR$(96);
360 PRINTCHR$(96);CHR$(96);CHR$(96);CHR$(96):PRINTCHR$(32);
380 PRINTCHR$(110);CHR$(109);CHR$(114);CHR$(110);CHR$(109)
400 PRINTCHR$(110);CHR$(32);CHR$(110);CHR$(125);CHR$(109);
420 PRINTCHR$(32);:PRINTCHR$(109)
440 PRINTCHR$(32);CHR$(110);CHR$(32);CHR$(125);CHR$(32);CHR$(109);
460 PRINTCHR$(32);CHR$(109)
480 PRINTCHR$(110);CHR$(32);CHR$(32);CHR$(125);CHR$(32);CHR$(32);
500 PRINTCHR$(109)
520 PRINTCHR$(32);CHR$(32);CHR$(32);CHR$(125)
```

Welche Zeichen sich hinter den Zahlen verstecken, können Sie in der Tabelle ab S. 122 nachsehen. Haben Sie das Programm richtig eingegeben, sollte diese Sonne auf dem Bildschirm erscheinen:

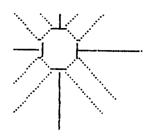


Abb. 1.2 zu Beispiel 1.2

Sicherlich ist zunächst gar nicht klar, welchen Nutzen eine solche Darstellung haben könnte. Wir wollen uns hier mit einer Andeutung begnügen: Im Unterschied zu den Zeichen selbst kann man mit den Zeichen-Codes rechnen, z.B. man kann mit einer einfachen Programm-Schleife den Zeichen-Code des Computers ganz oder teilweise ausgeben lassen. So liefert etwa die folgende BASIC-Anweisung die Großbuchstaben von 'A' bis 'Z' (d.h. die Zeichen mit den ASCII-Codes 65 bis 90):

```
FOR N = 65 TO 90: PRINT CHR$(N): NEXT N
```

Die Grafik-Zeichen lassen sich durch das folgende kurze Programm auf den Bildschirm bringen:

```
10 PRINT"□"
20 FOR I=96TO127
30 PRINT CHR$(I);
40 NEXT I
50 FOR I=161TO191
60 PRINT CHR$(I);
70 NEXT I
```

Wenn Sie in den Zeilen 30 und 60 zusätzlich den Wert der Variablen I ausgeben lassen, können Sie außerdem die jeweilige Code-Zahl der Zeichen sehen.

1.3. Grafik mit POKE-Anweisungen. Erklärung der POKE- und PEEK-Funktionen.

Um die nächste Möglichkeit der Grafik-Erstellung zu verstehen, muß man schon etwas tiefer in das "Rechnerinnenleben" einsteigen. Ein Computer besteht ja unter anderem aus vielen Speicherplätzen (der Commodore 64 beispielsweise aus mehr als 64000), die nur zu einem Teil für individuelle Daten und Programme des Benutzers frei zur Verfügung stehen. (Die Systemmeldung nach dem Einschalten des Rechners gibt die für den BASIC-Anwender bereitstehende Anzahl an Speicherplätzen an.) Der andere Teil wird vom Computer für ganz bestimmte Aufgaben benötigt: gewissermaßen für die 'persönliche Buchhaltung' des Computers. (Kapitel 7 enthält eine Übersicht über diese Speicherplätze.)

Damit man einen Computer beispielsweise überhaupt in BASIC (oder einer anderen "höheren" oder "problemorientierten" Programmiersprache wie z.B. Pascal oder FORTRAN) programmieren kann, müssen die Anweisungen aus einer solchen Sprache in die sogenannte "Maschinensprache" des Computers übersetzt werden, denn der Computer "versteht" letztlich nur einen ganz bestimmten Befehlssatz, den er gewissermaßen "fest eingebaut" hat.

Für jede Programmiersprache braucht ein Computer also einen anderen "Übersetzer". Das (in ROM-Speichern) eingebaute BASIC-Übersetzungs-Programm ist ein sogenannter "Interpreter". In dem Fall wird ein Programm nicht zuerst vollständig übersetzt und dann ausgeführt, sondern jede Anweisung wird sofort nach der Übersetzung ausgeführt, und zwar jedesmal von neuem! Wenn also z.B. eine Programmschleife 1000-mal durchlaufen wird, so werden alle in der Schleife stehenden Anweisungen 1000-mal übersetzt.

Der Vorteil eines solchen Interpreters besteht darin, daß sich Programme auf diese Weise sehr leicht und schnell im Dialog entwickeln und verändern lassen. Der Nachteil liegt in der geringeren Ablaufgeschwindigkeit von interpretierten BASIC-Programmen. Darauf wird in Kapitel 3 noch eingegangen. (Es gibt aber auch BASIC-Compiler, also Übersetzer, die das ganze Programm vor der Ausführung übersetzen).

Die meisten Speicherplätze eines Computers kann man nun direkt und einzeln ansprechen, und damit die Aufgabe steuern, für die ein Speicherplatz zuständig ist. (Ausgenommen davon ist der Interpreter, damit es nicht durch ein Versehen des Programmierers dazu kommt, daß der Computer seine BASIC-Kenntnisse "vergißt" und dann nur noch in Maschinensprache zu programmieren ist - das wäre meistens doch zu kompliziert.)

Um nun einen Speicherplatz zu veranlassen, seine Aufgabe auf eine ganz bestimmte Art und Weise auszuführen, verändert man den Wert, den dieser Speicherplatz normalerweise besitzt; man setzt einen neuen Wert in diesen Speicher. Die BASIC-Anweisung, mit der dies geschieht,lautet allgemein

```
POKE X,Y
```

Dabei ist X die Nummer (oder "Adresse") des Speicherplatzes, der angesprochen werden soll, und Y ist die Codenummer, die in diesen Speicherplatz geschrieben wird. In vielen Fällen kann man Y als eine für den Computer verschlüsselte Aufgabe interpretieren, die besagt, was genau dieser Speicher tun soll.

Probieren Sie doch einmal die beiden folgenden Anweisungen aus:

```
POKE 53280,0
POKE 53281,1
```

Damit verändern Sie die Rahmen-und Schriftfarbe zu schwarz, während der Bildschirmhintergrund weiß wird. Der Speicher 53280 kontrolliert also die Rahmen-, der Speicher 53281 die Hintergrundfarbe. Je nachdem, welche Werte Sie in diese Speicher hineinschreiben, können Sie beliebige Farbkombinationen von Rahmen und Hintergrund erzeugen. (Die Tabelle mit den Code-Nummern der Farben finden Sie auf S. 116).

1.3.1. Funktion und Verwendungsmöglichkeiten des Bildschirmspeichers

Für die Art der Grafik, die bisher behandelt wurde, sind zwei Gruppen von Speicherplätzen besonders wichtig. Die erste Gruppe umfaßt die Speicher-Adressen 1024 - 2033, das sind die Bildschirmspeicher. Mit den Speicherplätzen dieser Gruppe kann man steuern, an welcher Stelle des Bildschirms ein Zeichen erscheinen soll. Jeder Speicherplatz ist dabei für eine bestimmte Stelle des Bildschirms zuständig. Da der Bildschirm unterteilt wird in 25 Zeilen mit je 40 Spalten, läßt sich die Zuständigkeit der Speicher am besten so darstellen (vgl. auch die Tabellen in Kapitel 7):

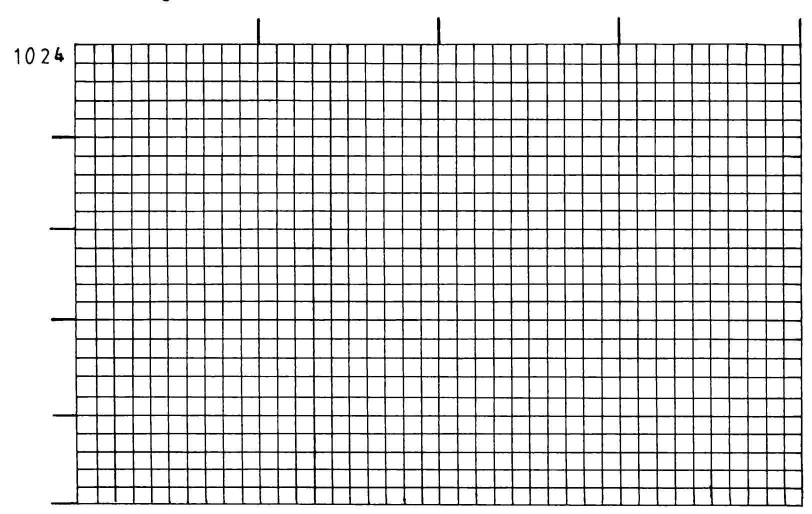


Abb. 1.3: Der Bildschirmspeicher

1024 ist die Speicher-"Adresse" des oberen linken Bildschirmpunktes. Mit Hilfe der Bildschirmspeicher kann man also genau angeben, an welcher Stelle ein Zeichen auf dem Bildschirm erscheinen soll. (Das ist mit einem einfachen PRINT-Befehl, wie Sie gesehen haben, wesentlich schwieriger!). Selbstverständlich muß man dem Computer auch noch angeben, welches Zeichen an der gewünschten Stelle stehen soll.

Will man jetzt in einen dieser Speicher einen Wert eintragen, braucht man noch die Code-Zahlen der Zeichen, die auf dem Bildschirm erscheinen können. (Diese Codes stehen in der Tabelle in Kapitel 7). Durch POKE 1024,81 kann man also den geschlossenen Kreis (Code 81) in der linken oberen Ecke des Bildschirms (Speicher 1024) erscheinen lassen.

Wie Sie vielleicht schon in der Tabelle im Handbuch auf Seite 133 gesehen haben, kann die Zahl 81 aber auch ein anderes Zeichen darstellen, nämlich den Buchstaben 'Q'. Entsprechendes gilt für die übrigen Codenummern, d.h. der Computer verfügt über zwei verschiedene Zeichensätze, wobei die einander entsprechenden Zeichen intern zwar durch die gleiche Codenummer gespeichert sind, auf dem Bildschirm aber beipielsweise entweder der Buchstabe 'Q' oder das Kreiszeichen erscheint. Um dem Computer mitzuteilen, welche dieser beiden Darstellungen gewünscht wird, kann man im Programm durch die Anweisung POKE 53272,23 auf den zweiten Zeichensatz umschalten. Mit POKE 53272,21 gelangt man zurück in den ersten Zeichensatz, der auch der normalen Darstellung auf dem Bildschirm (Großbuchstaben und Commodore-Grafik-Zeichen) entspricht.

Wenn Sie den Befehl POKE 1024,81 jetzt ausprobieren wollen, sehen Sie allerdings nur dann eine Veränderung, wenn schon irgendetwas in der linken oberen Ecke steht, bevor Sie den POKE-Befehl geben. Das gePOKEte Zeichen erscheint nämlich in der gleichen Farbe wie das Zeichen, das vorher an der entsprechenden Stelle stand. Befand sich dort nichts, erhält das gePOKEte Zeichen die Hintergrundfarbe und bleibt damit unsichtbar.

1.3.2. Erläuterung des Farbspeichers. Zusammenhang von Bildschirm- und Farbspeicher.

Um die gewünschten Zeichen nun auch z.B. auf einem leeren Bildschirm sichtbar zu machen, muß man noch über eine zweite Speichergruppe Bescheid wissen. Dabei handelt es sich um die Speicher mit den Adressen 55296 bis 56295, die Farbspeicher. Diese sind zuständig für die Farbe, die ein Zeichen, das an einer bestimmten Stelle steht, haben soll. In diese Speicher werden also Codes für Farben gePOKEt. Die Aufteilung entspricht der des Bildschirmspeichers (s. auch Kapitel 7):

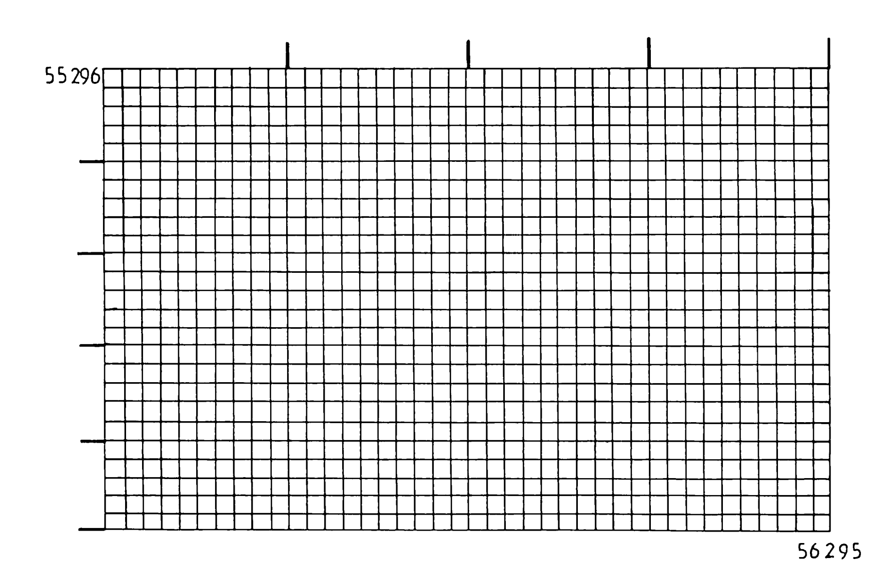


Abb. 1.4: Der Farbspeicher

Probieren Sie doch einmal folgendes Programm:

```
10 REM.................................BEISPIEL 1.3
20 PRINT"♥"
30 PRINT"REGENBOGEN"
40 REM.................................FARBEN
50 POKE 55336,4:POKE 55337,5:POKE 55338,13:POKE 55339,15
60 POKE 55340,1:POKE 55341,7:POKE 55342,8:POKE 55343,10
70 POKE 55344,5:POKE 55345,13
```

Um die Zeichen, deren Codes man in den Bildschirmspeicher (1024 - 2023) geschrieben hat, auch sehen zu können, muß man also in der Farbspeicher-Tabelle die Plätze (d.h. die Speicheradressen) suchen, die den Stellen in der Bildschirmspeicher-Tabelle entsprechen und in diese Speicherzellen den Code der gewünschten Farbe hineinschreiben (mit POKE).

Geben Sie also die beiden Befehle POKE 1024,81 und POKE 55296,7, so erscheint ein gelber Kreis in der linken oberen Ecke des Bildschirms.

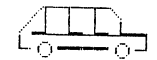
Mit Hilfe des POKE-Befehls sowie des Bildschirm- und Farbspeichers wird nun das nächste Beispiel erstellt:

```
10 REM................................BEISPIEL 1.4
20 PRINT"□"
850 REM...............................FARBE AUTO
860 FOR J=0TO3:Q=55984+J*40
870 FOR I=0TO10:POKEQ+I,3
880 NEXT I
890 NEXT J
895 REM...............................BESTANDTEILE AUTO
900 POKE 1714,78:POKE 1715,79:POKE 1716,80
920 POKE 1717,119:POKE 1718,80:POKE 1719,119
940 POKE 1720,77:POKE 1753,78:POKE 1754,111
960 POKE 1755,76:POKE 1756,122:POKE 1757,111
980 POKE 1758,122:POKE 1759,111:POKE 1760,122
1000 POKE 1761,111:POKE 1762,111
1020 POKE 1793,76:POKE 1794,85:POKE 1795,73
1040 POKE 1796,111:POKE 1797,111:POKE 1798,111
1060 POKE 1799,111:POKE 1800,85:POKE 1801,73
1080 POKE 1802,122:POKE 1834,74:POKE 1835,75
1120 POKE 1840,74:POKE 1841,75
```

Die Zeilen 860 und 870 sorgen für die Farbe. Der Einfachheit halber wird hier in der Farbspeicher-Tabelle ein Rechteck ausgewählt, das sämtliche vorkommenden Bildschirmspeicherplätze umfaßt.

Dieses Programm können Sie natürlich auch einfach an die vorigen Beispiel-Programme anhängen, um alle Darstellungen gleichzeitig zu sehen! (Die Zeilen sind entsprechend numeriert.)

Mit den folgenden Programmzeilen können Sie das Haus auch noch farbig gestalten, wobei einzelne Teile verschiedene Farben bekommen können:



```
10 REM...............................BEISPIEL 1.5
1160 REM..............................FARBE HAUS
1170 FOR J=0TO5:Q=55922+J*40
1180 FOR I=0TO12:POKEQ+I,8
1190 NEXT I
1200 NEXT J
1210 REM..............................FENSTER
1220 FOR J=0TO2:Q=56003+J*40
1230 FOR I=0TO10:POKEQ+I,5
1240 NEXT I
1250 NEXT J
1255 REM..............................DACH
1260 FOR J=0TO6:Q=55642+J*40
1270 FOR I=0TO12:POKEQ+I,2
1280 NEXT I
1290 NEXT J
1295 REM..............................SCHORNSTEIN
1300 POKE 55771,8:POKE 55772,8:POKE 55811,8:POKE 55812,8
1310 REM..............................RAUCH
1320 FOR J=0TO1:Q=55691+J*40
1330 FOR I=0TO3:POKEQ+I,1
1340 NEXT I
1350 NEXT J
1355 REM..............................FARBE SONNE
1360 FOR J=0TO7:Q=55296+J*40
1370 FOR I=0TO9:POKEQ+I,7
1380 NEXT I
1390 NEXT J
```

Die bisher vorgestellten Beispiele sollten zeigen, wie man den Bildschirm- und den Farbspeicher anspricht und wie beide im Zusammenhang benutzt werden können.

1.4. Bewegte Grafik mit den "fest eingebauten" Grafik-Zeichen.

Mit dem folgenden Programm wird aus den Grafik-Zeichen ein Strich-Männchen zusammengesetzt.

```
1399 REM.................................BEISPIEL 1.6
1400 FOR I=56136TO56295:POKEI,10:NEXT I:REM FIGUR
1420 R=1864
1440 :
1460 FOR I=0TO3:Q=R+I*40:REM.............FIGUR SETZEN
1480 READ A,B:POKE Q,A:POKE Q+1,B:NEXT I
1570 :
2000 DATA 85,73,74,75,80,79,78,77
```

In diesem Programm-Beispiel wird nun nicht mehr, wie beim "Auto", jeder betroffene Speicherplatz einzeln angesprochen. Da die Plätze, die für das Strich-Männchen nötig sind, zusammenhängen, wurde stattdessen eine Schleife verwendet.

Die Code-Nummern der Zeichen, aus denen sich die Figur aufbaut, stehen in der DATA-Zeile. Findet der Computer in Zeile 2000 die READ-Anweisung, liest er aus der DATA-Zeile die Werte für die Variablen A und B aus der READ-Anweisung. Beim nächsten Schleifendurchgang holt er sich die beiden nächsten Werte aus der DATA-Zeile usw.

Soll sich das Strich-Männchen auch noch bewegen, muß das Programm etwas erweitert werden:



```
10 REM..................................BEISPIEL 1.7
20 PRINT"〕"
1400 FOR I=56136TO56295:POKE I,10:NEXT I:REM FARBE FIGUR
1420 R=1864
1440 :
1450 FOR J=0TO38
1460 FOR I=0TO3:Q=R+I*40:REM..............FIGUR SETZEN
1480 READ A,B:POKEQ+J,A:POKEQ+J+1,B:NEXTI:RESTORE
1500 FOR Z=1TO250:NEXTZ:REM...............WARTESCHLEIFE
1520 FOR I=0TO3:Q=R+I*40:REM..............FIGUR LOESCHEN
1540 POKE Q+J,96:POKE Q+J+1,96:NEXT I
1560 NEXT J
1570 :
1580 GOTO 1440
2000 DATA 85,73,74,75,80,79,78,77
```

Die Bewegung entsteht dabei so, daß das Männchen an einer Stelle zusammengesetzt wird, dann an dieser Stelle gelöscht und eine Position weiter wieder aufgebaut wird. (Das Löschen erfolgt dadurch, daß der Wert 0 in die betreffenden Bildschirmspeicherplätze geschrieben wird.) Um das wiederholte Zeichnen zu ermöglichen, muß der RESTORE-Befehl (in Zeile 1480) verwendet werden. Er weist den Computer an, die Daten wieder von vorne zu lesen. Zwischen dem Zusammensetzen und dem Löschen der Figur steht eine Warteschleife, damit der ganze Vorgang nicht zu schnell abläuft. Indem Sie diese Schleife verändern, können Sie das Männchen schneller oder langsamer laufen lassen.

Wie Sie sehen, bewegt sich das Männchen etwas ruckartig. Dafür gibt es eine einfache Erklärung. Die einzelnen Zeichen dieser Figur werden bei jedem Schritt nacheinander gelöscht und nacheinander wieder aufgebaut. Diese zeitliche Verzögerung läßt die Bewegung ruckartig erscheinen.

Eine wesentlich bessere Darstellung solcher Figuren - sowohl in Bezug auf die Grafikauflösung, als auch in Bezug auf die Bewegungen - erhält man mit einer anderen Grafikmöglichkeit des Commodore 64, die im nächsten Kapitel vorgestellt wird.

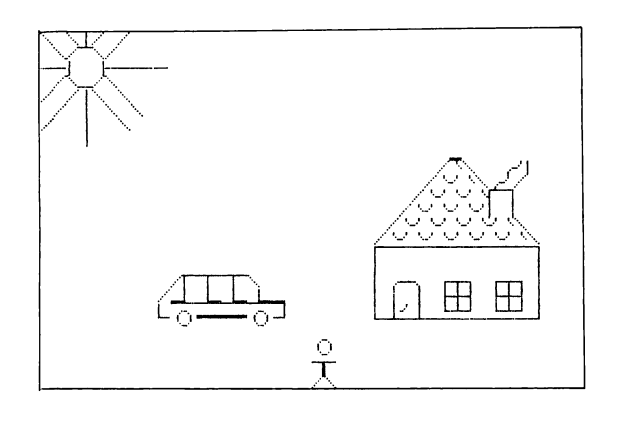


Abb. 1.5 zu Beispiel 1.1 bis 1.7

1.5. Z u s a m m e n f a s s u n g

Mit den auf der Tastatur vorhandenen Grafikzeichen kann man auf verschiedene Arten grafische Darstellungen auf den Bildschirm bringen:

1. Direkte Erzeugung von Bildschirm-Grafiken. Man kann mit Hilfe der Cursor-Steuerung einfach direkt auf dem Bildschirm Muster oder Zeichnungen zusammensetzen. Verschiedene Farben erhält man durch gleichzeitiges Drücken der CTRL- oder der Commodore-Taste und einer Farbtaste (d.h. einer Zifferntaste von 1 bis 8).

Diese Zeichnungen können aber nicht "aufbewahrt" werden. Will man eine Grafik öfter verwenden, muß man ein Programm zur grafischen Darstellung schreiben.

2. Erzeugung von Bildschirm-Grafiken durch Programmme. Dazu gibt es drei Möglichkeiten:

2.1 Die einfachste Möglichkeit zur Grafik-Erzeugung durch ein Programm besteht darin, die Zeichen, die in einer Bildschirmzeile stehen sollen, einfach in Anführungszeichen hinter eine PRINT-Anweisung zu schreiben.

2.2 Für den PRINT-Befehl kann man aber auch die Zeichen durch den jeweiligen CHR$-Code verschlüsseln und diese Werte dann z.B. alle hintereinander in DATA-Zeilen angeben, aus denen sie durch wiederholte READ-Anweisungen herausgelesen und die entsprechenden Zeichen auf dem Bildschirm ausgegeben werden können.

2.3 Die dritte Methode ist die Benutzung des POKE-Befehls unter Verwendung des Bildschirm- und Farbspeichers. Hierbei verändert man direkt die Werte in den entsprechenden Rechnerspeichern. Das hat den Vorteil, daß jedes Zeichen in einer gewünschten Farbe immer an einer ganz genau festgelegten Stelle des Bildschirms erscheint. (Mit der PRINT-Anweisung erreicht man dies nur, wenn zu Beginn des Programmes der Bildschirm gelöscht wird - mit: PRINT "♥").

Die Verwendung der Commodore-Grafik-Zeichen hat aber oft den Nachteil, daß die Darstellung zu grob ist. Eine feinere Aufteilung erhält man durch die "Sprites" (das sind selbsterstellte bewegliche grafische Objekte) und durch die "hochauflösende Grafik".

2. Sprites

"Sprites" (dt.: Kobolde) sind vom Benutzer frei definierbare Figuren. Sie sind damit nicht mehr festgelegt durch die Möglichkeiten, die die Grafik-Zeichen der Tastatur bieten! Außerdem lassen sich Sprites, wenn sie einmal definiert sind, sehr einfach im ganzen Bildschirmbereich bewegen.

Sprites werden in hochauflösender Grafik dargestellt. Das bedeutet, daß jeder Bildschirmpunkt der normalen Darstellung (also der Platz, den z.B. ein Buchstabe oder der Cursor einnimmt) weiter unterteilt wird in je 64 Einzelpunkte (8 Zeilen mit je 8 Spalten). Der gesamte Bildschirm besteht dann also aus 40 x 8 (= 320) Punkten pro Zeile mal 25 x 8 (= 200) Punkten pro Spalte, insgesamt 64000 Einzelpunkten. In diese winzigen Punkte können Sie natürlich keine Buchstaben oder Commodore-Grafik-Zeichen mehr schreiben. Hier können Sie dem Rechner nur noch angeben, ob ein Punkt gesetzt werden soll oder nicht. Dies geschieht über POKE-Befehle. Näheres dazu erfahren Sie sowohl in diesem Kapitel als auch in dem darauf folgenden über die hochauflösende Grafik.

Eine wichtige Einschränkung gilt es noch zu beachten: Sprite-Figuren können nicht beliebig groß sein. Jedes Sprite setzt sich aus einem Punkte-Raster von 24 Punkten (in der Breite) x 21 Punkten (in der Höhe) zusammen. Außerdem können höchstens 8 Sprites gleichzeitig auf dem Bildschirm erscheinen.

Bevor Sie nun Ihre eigenen Entwürfe auf dem Bildschirm sehen können, müssen Sie zunächst die für Sprites zuständigen Speicherplätze des Rechners kennen. Außerdem ist es notwendig, ein bestimmtes Entwurfsverfahren einzuhalten, um sich dem Rechner "verständlich" machen zu können.

Die Einzelheiten in diesem Kapitel mögen Ihnen zuerst recht kompliziert vorkommen. Nachdem Sie aber einen Überblick über die zuständigen Speicherbereiche des Computers gewonnen haben, werden Ihnen die - leider notwendigen - zahlreichen POKE-Befehle verständlich werden. Dazu werden nützliche Hilfsmittel vorgestellt, die einige Dinge vereinfachen.

2.1. Entwurf von Sprite-Figuren.

Um eine Sprite-Figur zu entwerfen, benötigen Sie zunächst einmal ein Raster in der entsprechenden Größe (24 x 21 Punkte). Sie können sich dieses Raster natürlich selbst herstellen oder aber das beigefügte Sprite-Entwurfsblatt benutzen. In dieses Raster zeichnen Sie jetzt die gewünschte Figur ein. Anschließend müssen Sie jedes Kästchen, das von der Zeichnung berührt wird, ausfüllen. (Wir haben als Beispiel eine Lokomotive gewählt).

Damit der Computer diese Figur auch darstellen kann, muß jetzt in verschlüsselter Form angegeben werden, welche Kästchen ausgefüllt sind und welche nicht. Diese Daten geben dem Rechner an, welche Bildschirmpunkte zu setzen sind und welche nicht.

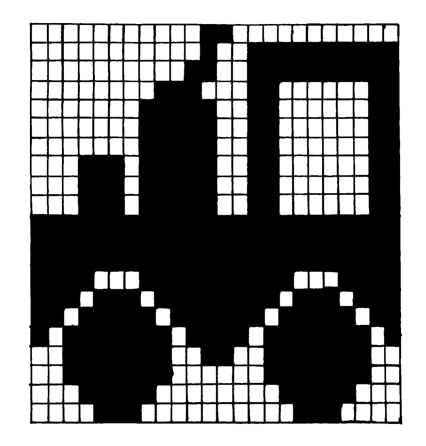
Zur Berechnung der Daten schreiben Sie über Ihr Sprite-Raster von links nach rechts dreimal hintereinander die Zahlen 128 64 32 16 8 4 2 1 über die obere Kästchenreihe. (Beim Sprite-Entwurfsblatt haben wir Ihnen diese Arbeit abgenommen). Für jeweils eine Gruppe dieser Zahlen werden nun die Werte addiert, die über <u>ausgefüllten</u> Kästchen stehen. Damit erhält man die Daten für den Commodore-Computer: je Sprite-Zeile also 3 und für das gesamte Sprite 63 Werte.

Machen Sie sich dieses Rechenschema am besten an unserem Beispiel klar:

In der ersten Gruppe der ersten Zeile ist kein Kästchen ausgefüllt, der Datenwert lautet also 0. In der zweiten Gruppe dieser Zeile sind die Kästchen unter den Zahlen 16 und 8 ausgefüllt, wir erhalten also: 16 + 8 = 24. Für die dritte Gruppe der ersten Zeile ergibt sich wieder 0. Auf diese Weise berechnen Sie pro Sprite-Zeile 3 Werte, für das gesamte Sprite also 63.

Das ausgefüllte Entwurfsblatt mit den Daten sollte nach Abschluß der Berechnungen folgendermaßen aussehen:

SPRITE-ENTWURF



0	24	0
0	19	255
0	51	255
0	227	3
1	243	3
1	243	3
1	243	3
29	243	3
29	243	3
29	243	3
255	255	255
255	255	255
255	255	255
241	255	143
238	255	119
223	126	251
191	189	252
63	153	252
63	129	252
31	0	248
14	0	112

Abb. 2.1

(Sollten Ihnen diese Berechnungen viel zu mühselig sein, um sie öfter als einmal auszuprobieren: in Kapitel 5 finden Sie ein Hilfsprogramm zur Sprite-Erstellung, das Ihnen insbesondere diesen Rechenaufwand abnimmt. Arbeiten Sie aber trotzdem zuerst das vorliegende Kapitel durch, damit Sie das Programmieren von Sprites besser verstehen lernen!)

2.2. Darstellung von Sprites auf dem Bildschirm.

Was fängt man nun mit diesen Zahlen an? Um sich die Figur auf dem Bildschirm anzusehen, genügt folgendes Programm:

```
10 REM..................................BEISPIEL 2.1
20 PRINT"□"
50 V=53248:REM..........................VIDEOCHIP
100 POKE V+21,4:REM.....................LOK
150 POKE 2042,13:REM....................BLOCK NR.
200 FORI=0TO62:READ Q:POKE 832+I,Q:NEXTI:REM DATEN EINLESEN
240 REM.................................KOORDINATEN LOK
250 POKE V+4,200
300 POKE V+5,100:REM.......KOORDINATEN
400 REM.................................DATEN LOK
500 DATA 0,24,0,0,19,255,0,51,255,0,227,3,1,243,3,1
510 DATA 243,3,1,243,3,29,243,3,29
520 DATA 243,3,29,243,3,255,255,255,255,255
530 DATA 255,255,255,255,241,255,143,238,255
540 DATA 119,223,126,251,191,189,252,63,153
550 DATA 252,63,129,252,31,0,248,14,0,112
```

(Beim Eintippen der Daten kann man sich leicht vertun. Achten Sie darauf, daß in den DATA-Zeilen auch tatsächlich 63 Werte stehen.)

... und das tut dieses Programm:

Zeile 50: 53248 ist die Adresse des ersten Speichers der Gruppe der Speicherplätze, die für die hohe Auflösung des Bildschirms zuständig sind. Diese Zahl wird unter V gespeichert, damit die Adresse nicht ständig wiederholt werden muß.

Zeile 100: 21 Speicherplätze weiter (V+21) liegt der Speicher, der ein Sprite überhaupt erscheinen läßt.

Zeile 150: Hier wird dem Computer gesagt, woher er die Daten für das Sprite herholen soll.

Zeile 200: Hier werden die Werte aus den DATA-Zeilen gelesen und nacheinander in die Speicherzellen gePOKEt, die der Angabe in der vorigen Zeile entsprechen.

Zeile 250: Angabe der X-Koordinate der Stelle, an der das Sprite erscheinen soll. (X-Koordinate: Verschiebung in einer Bildschirmzeile nach rechts.)

Zeile 300: Angabe der Y-Koordinate (Verschiebung in einer Bildschirmspalte nach unten).

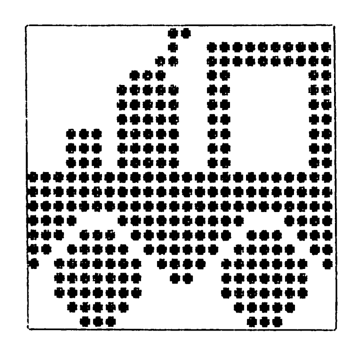


Abb. 2.2 zu Beispiel 2.1

Diese Erklärungen sind natürlich noch viel zu ungenau, um damit wirklich arbeiten zu können. Welcher Wert muß denn nun in welchen Speicher geschrieben werden? Die Speicherstellen ab V (53248) haben folgende Funktionen:

V+0	Bestimmung der X-Koordinate von Sprite Nr. 0.
V+1	Y-Koordinate von Sprite Nr. 0.
V+2-V+15	Diese Sprites kontrollieren jeweils paarweise die X- und Y-Koordinaten für die Sprites Nr. 1 bis 7.
V+16	X-Koordinate zwischen 256 und 320.
V+21	Läßt Sprites erscheinen oder löscht Sprites.
V+23	Verdoppelt ein Sprite in Y-Richtung.
V+27	Setzt Sprite in den Vorder- oder Hintergrund.
V+29	Verdoppelt ein Sprite in X-Richtung.
V+30	Registriert Zusammenstoß von Sprites.
V+31	Registriert Zusammenstoß von Sprite und Hintergrund.
V+37,V+38	Farbspeicher für mehrfarbige Sprites.
V+39-V+46	Farben der Sprites Nr. 0 - 7.

V+21:

Der wichtigste Speicherplatz in dieser Liste ist V+21: dieser Speicher hat die Funktion eines Schalters, der die Sprite-Darstellung ein- oder ausschaltet. In diesen Speicher muß nun die Code-Zahl des Sprites geschrieben werden, das man auf dem Bildschirm sehen möchte. Um den Code-Wert zu ermitteln, kann man sich vorstellen, daß ein Speicherplatz in sich wieder unterteilt ist in 8 Stellen, von denen jede für eins der 8 möglichen Sprites (0 - 7) zuständig ist:

128	64	32	16	8	4	2	1	Code-Zahlen
7	6	5	4	3	2	1	0	Sprite-Nr.

Die Nummer jedes Sprites, das erscheinen soll, wird in der unteren Reihe angekreuzt. Die diesen Kästchen entsprechenden Code-Zahlen der oberen Reihe werden addiert. Möchte man also Sprite Nr. 2 auf den Bildschirm bringen, muß man eine 4 in den Speicher V+21

POKEn (Zeile 100 im Beispiel-Programm). Sollen die Sprites Nr. 3 und Nr. 6 erscheinen, lautet der Befehl entsprechend: POKE V+21,72 (72 = 64 + 8 ; 64: Code-Zahl von Sprite Nr. 6; 8: Code-Zahl von Sprite Nr. 3).

<u>2040 - 2047:</u>

Außerdem muß der Computer wissen, in welchen Speicherplätzen die Daten für das gewünschte Sprite abgespeichert sind. Für diese Angabe sind die Speicher Nr. 2040 bis 2047 zuständig. In 2040 wird angegeben, wo die Werte für Sprite Nr. 0 stehen, in 2041 entsprechend für Sprite Nr. 1 usw. Um den zu POKEnden Wert zu erhalten, muß man wissen, daß die Speicher in Blöcken von je 64 Speicherplätzen zusammengefaßt werden. Da man zur Definition eines Sprites ja 63 Daten braucht, passen die Daten für ein Sprite in einen Block. In den entsprechenden Speicher von 2040 bis 2047 wird also die Nummer des Blocks geschrieben, in dem die Daten für das gewünschte Sprite stehen sollen.

In dem Beispielprogramm sollte Sprite Nr. 2 auf dem Bildschirm erscheinen (V+21,4). Zu Sprite Nr. 2 gehört hier der Speicher 2042. POKE 2042,13 bedeutet also für den Computer: die Daten für Sprite 2 sind aus dem 13. Block von Speicherstellen zu holen.

Als mögliche Speicherbereiche, in denen die Sprite-Daten stehen können, stehen die Blocks Nr. 10 bis Nr. 20 zur Verfügung. Die ersten Speicherplätze jedes Blocks haben folgende Adressen:

Block	1.Speicherplatz
10	640
11	704
12	768
13	832
14	896
15	960
16	1024
17	1088
18	1152
19	1216
20	1280

Die Anfangsadressen der Blocks errechnen sich aus Block-Nummer (z.B. 16) x Blocklänge (64 Speicherplätze). Für Block Nr. 16 erhält man also als ersten Speicherplatz den Speicher Nr. 1024.

Noch stehen die Sprite-Daten aber nicht im 13. Block. Dahin werden sie in der nächsten Zeile gebracht:

```
FOR I = 0 TO 62: READ Q: POKE 832+I,Q: NEXT I
```

Hier werden nacheinander insgesamt 63 Werte (Angaben für ein Sprite) aus den DATA-Zeilen gelesen und in die 63 Speicherplätze von 832 bis 894 gePOKEt. 832 ist dabei der erste Speicherplatz des 13. Blocks.

V+4 und V+5:

Jetzt muß der Rechner nur noch wissen, wo das Sprite erscheinen soll. Dazu dienen die nächsten beiden Zeilen. POKE V+4,160 bedeutet: Sprite Nr. 2 soll 160 Punkte vom linken Rand entfernt auftauchen. POKE V+5,100 gibt an, daß dieses Sprite 100 Punkte vom oberen Rand entfernt sein soll. Im Kreuzungspunkt der entsprechenden Zeile und Spalte erscheint dann das Sprite.

2.3. Bewegung von Sprites.

Sind Sie bis hierher gekommen, ist es jetzt sehr einfach, die Sprites zu bewegen oder ihre Größe zu verändern.

Verändern Sie das Beispielprogramm 2.1 doch einmal folgendermaßen:

```
10 REM..................................BEISPIEL 2.2
20 PRINT"♥"
40 V=53248:REM..........................VIDEOCHIP
120 POKE 2042,13:REM....................BLOCK NR.
140 FOR I=0TO62:READ Q:POKE 832+I,Q:NEXT I:REM DATEN EINLESEN
200 POKE V+21,4:REM.....................LOK
210 FOR X=255TO0STEP-.25:REM............BEWEGUNG
220 POKE V+4,X:POKE V+5,160:REM.........KOORDINATEN
270 NEXT X
390 GOTO210:REM.........................BEWEGUNG WIEDERHOLEN
400 REM.................................DATEN LOK
500 DATA 0,24,0,0,19,255,0,51,255,0,227,3,1,243,3,1
510 DATA 243,3,1,243,3,29,243,3,29
520 DATA 243,3,29,243,3,255,255,255,255,255
530 DATA 255,255,255,255,241,255,143,238,255
540 DATA 119,223,126,251,191,189,252,63,153
550 DATA 252,63,129,252,31,0,248,14,0,112
```

Da die X-Koordinate (angegeben durch V+4) durch die Schleife laufend ihren Wert verändert, die Y-Koordinate aber feststeht, bewegt sich die Lok jetzt auf einer waagerechten Linie von rechts nach links über den Bildschirm.

Durch Ändern der Schrittweite für die Schleife (z.B.: FOR X = 0 TO 255 STEP 5 oder FOR X = 0 TO 255 STEP 0.25) kann man außerdem die Geschwindigkeit steuern (z.B. STEP 5: schneller, STEP 0.25: langsamer).

2.4. Weitere Möglichkeiten der Sprites-Grafik, z.B.: Farbe, Vergrößern, Bewegung über den ganzen Bildschirm.

Mit diesen Angaben kann man also schon eigene Figuren auf dem Bildschirm sich bewegen lassen. Im folgenden werden die übrigen Speicherstellen besprochen, die mit den Sprites zu tun haben.

V+16:

Mit dem Speicher V+16 erreicht man eine Bewegung über den ganzen Bildschirm. Da der Bildschirm für die

Sprites in 320 Punkte pro Zeile aufgeteilt wird, man in jede Speicherzelle des Rechners aber nur Werte zwischen 0 und 255 POKEn kann (Erklärung dazu s. Kapitel 4) läßt sich ein Sprite nie ganz an den rechten Bildschirmrand setzen. Um dies dennoch zu ermöglichen, kann man in den Speicher V+16 die Code-Zahl für das (oder die) Sprite(s), das (die) weiter als 255 Punkte nach rechts verschoben werden soll(en), POKEn. Diese Code-Zahl errechnet sich wieder wie für V+21 oder V+23 bzw. V+29. Für Sprite 2 heißt das also: POKE V+16,4. Anschließend POKEt man in V+4 (X-Koordinate für Sprite Nr. 2) eine Zahl zwischen 0 und 63 und erreicht so die Punkte 256 bis 319.

V+23 und V+29:

Der Speicherplatz V+23 bewirkt eine Vergrößerung des Sprites in Y-Richtung. Dazu muß die Code-Zahl des Sprites, das vergrößert werden soll, in diesen Speicher hinein. Diese Code-Zahl wird genauso berechnet wie für den Speicher V+21. Will man also Sprite 2 vergrößern, lautet der Befehl: POKE V+23,4. Sollen gleichzeitig Sprite 2 und 3 vergrößert werden, gibt man an: POKE V+23,12. Auf die gleiche Art und Weise kann man Sprites auch in X-Richtung vergrößern (Speicherplatz V+29).

V+27:

Der Speicher V+27 kontrolliert, ob ein Sprite im Vorder- oder Hintergrund erscheinen soll. (Vorder- bzw. Hintergrund bezieht sich dabei auf Text oder Commodore-Grafik-Zeichen.) Um diesen Effekt auszuprobieren, stoppen Sie die Lok irgendwo und schreiben ein paar Zeichen vor und "über" die Lok. Sie werden feststellen, daß die Lok vor dem Text zu stehen scheint.

Gehen Sie nun in eine freie Zeile und geben den Befehl POKE V+27,4. Nun erscheint die Lok hinter dem Text. - In den Speicher V+27 müssen die Code-Zahlen derjenigen Sprites gePOKEt werden, die im Hintergrund erscheinen sollen. (Diese Codes berechnen sich wieder genauso wie für die für den Speicher V+21. Sollen mehrere Sprites im Hintergrund sein, werden ihre Werte entsprechend addiert.)

V+30:

Dieser Speicher kontrolliert, ob zwei oder mehrere Sprites zusammenstoßen. Der Wert dieses Speichers - Sie können Ihn mit PRINT PEEK(V+30) abfragen - ist solange 0, bis eine Kollision zwischen verschiedenen Sprites erkannt wird. Dann nimmt der Speicher den Wert an, der sich durch Addition der Code-Zahlen der zusammengestoßenen Sprites ergibt. (Die Code-Zahlen entsprechen wieder den von V+21 her bekannten Werten).

Nach einer Abfrage, ob z.B. Sprite Nr. 0 und Sprite Nr. 1 zusammengestoßen sind, ob also PEEK(V+30) gleich 3 ist, könnte man beispielsweise Sprite Nr. 0 löschen (indem der Wert in V+21 entsprechend verkleinert wird) oder irgendeine andere Aktion auslösen.

Nach einem Zusammenstoß muß dieser Speicher durch POKE V+30,0 wieder zurückgesetzt werden, sonst würden weiterhin die alten Sprites als zusammengestoßen registriert werden.

V+31:

Der Speicher V+31 kontrolliert den Zusammenstoß von Sprites mit dem Hintergrund, also etwa Zeichen auf dem Bildschirm. Man kann allerdings durch die Abfrage PEEK(V+31) nur erkennen, welches Sprite mit dem Hintergrund zusammengestoßen ist, nicht aber, mit welchem Zeichen. Nach einer Kollision nimmt der Speicher V+31 den Code-Wert des betroffenen Sprites an. Auch dieser Speicher muß wieder zurückgesetzt werden.

V+37 und V+38:

Diese beiden Speicher brauchen Sie, wenn Sie mehrfarbige Sprites entwerfen wollen. In beide Speicherplätze können Sie jeweils die Code-Zahl einer Farbe POKEn; diese Farben gelten dann für alle mehrfarbigen Sprites. Außerdem kann für jedes einzelne Sprite noch eine Farbe im entsprechenden Sprite-Farbspeicher (V+39 - V+46) bestimmt werden. Damit können Sie also dreifarbige Sprites auf dem Bildschirm erscheinen lassen.

Der Entwurf mehrfarbiger Sprites unterscheidet sich allerdings etwas vom Entwurf der einfarbigen.

Jetzt muß der Rechner ja nicht nur wissen, an welche Stelle ein Punkt gesetzt oder nicht gesetzt werden soll, sondern auch, in welcher Farbe ein gesetzter Punkt erscheinen soll. Diese zwei verschiedenen Informationen zu einem Sprite-Punkt werden im gewohnten Sprite-Entwurfsblatt (oder im Sprite-Entwurfs-Programm in Kapitel 5) durch zwei Kästchen festgelegt. Jetzt bestimmen also zwei Kästchen, wie ein Sprite-Punkt gesetzt werden soll. Um diese beiden Kästchen auszufüllen, gibt es vier verschiedene Möglichkeiten:

1. ☐☐ beide Kästchen leer: der Sprite-Punkt wird nicht gesetzt,

2. ☒☐ erstes Kästchen ausgefüllt: der Sprite-Punkt erscheint in der Farbe, die im entsprechenden Sprite-Farbspeicher (V+39 - V+46) steht,

3. ☐☒ zweites Kästchen ausgefüllt: der Sprite-Punkt bekommt die Farbe aus Speicher V+37,

4. ☒☒ beide Kästchen ausgefüllt: der Punkt wird in der Farbe aus Speicher V+38 gesetzt.

Als Beispiel haben wir einen mehrfarbigen 'Triebwagen' entworfen. Das Bild ist allerdings im Entwurf nicht mehr so eindeutig zu erkennen.

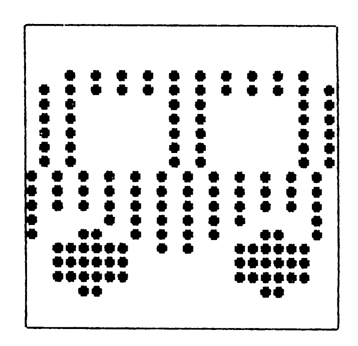


Abb. 2.3

```
10 REM..................................BEISPIEL 2.3
20 :
30 PRINT"□"
40 V=53248
50 POKE V+21,4: POKE V+28,4
60 POKE V+37,7: POKE V+38,0: POKE V+41,2: POKE2042,13
70 FOR I=0TO62: READ Q: POKE 832+I,Q: NEXT I
80 FOR X=0TO255
90 POKE V+4,X: POKE V+5, 180
100 NEXT X
110 GOTO 80
2000 DATA 0,0,0,0,0,0,0,0,0,21,85,84,85,85,84,80,20
2010 DATA 5,208,20,5,208,20,5,208,20,5,208,20,5
2020 DATA 170,170,170,170,170,170,170,170,170,130,170,130,140
2030 DATA 170,50,63,40,252,63,0,252,63,0,252,12,0,48,0,0,0,0,0,0
```

Diese Programmzeilen werden Sie im Eisenbahn-Programm in Kapitel 5 wiederfinden.

In Zeile 50 wird durch POKE V+28,4 angegeben, daß das Sprite Nr. 2 (Code-Zahl 4) mehrfarbig erscheinen soll.

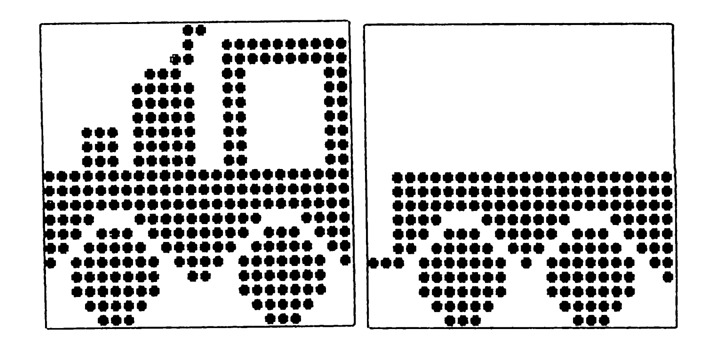
In Zeile 60 werden die drei möglichen Farben bestimmt: V+37 erhält den Wert 7 (gelb), V+38 den Wert 0 (schwarz) und in den Farbspeicher für Sprite Nr. 2 wird die Zahl 2 (für 'rot') geschrieben. Die übrigen Angaben kommen Ihnen wohl bekannt vor.

V+39 bis V+46:

Mit den Speichern V+39 bis V+46 können Farben für die einzelnen Sprites gewählt werden. Hier können also die Codes 1 - 16 (für die Farben) verwendet werden. V+39 ist für die Farbe von Sprite Nr. 0 zuständig, V+40 für Sprite Nr. 1 usw.

In den folgenden Beispielen werden Sie einige dieser erweiterten Möglichkeiten kennenlernen.

```
10 REM...............................BEISPIEL 2.4
20 PRINT"□"
40 V=53248:REM.......................VIDEOCHIP
120 POKE 2042,13:POKE 2043,14:POKE 2044,14:REM BLOCK NR.
140 FOR I=0TO62:READ Q:POKE 832+I,Q:NEXTI:REM DATEN EINLESEN
150 FOR I=0TO62:READ Q:POKE 896+I,Q:NEXTI:REM DATEN EINLESEN
160 POKE V+42,7:POKE V+43,2:REM.........FARBE WAGEN
180 POKE V+29,16:REM....................VERGROESSERUNG
200 POKE V+21,12:REM....................LOK UND WAGEN
210 FOR X=255TO0STEP-.25:REM............BEWEGUNG
220 POKE V+4,X:POKE V+5,160
240 IFX<=230THEN POKE V+6,X+25:POKE V+7,160
260 IFX<=205THEN POKE V+21,28:POKE V+8,X+50:POKE V+9,160
270 NEXT X
280 FOR X=25TO  0STEP-.25
290 POKE V+6,X:POKE V+7,160:POKE V+8,X+25
300 POKE V+9,160
310 FOR J=1TO10:NEXT J:REM...............WARTESCHLEIFE
320 NEXT X
330 FOR X=25TO  0STEP-.25
340 POKE V+8,X:POKE V+9,160
350 IFX=0THEN POKEV+21,0
360 FOR J=1TO20:NEXTJ:REM................WARTESCHLEIFE
370 NEXT X
380 POKE V+21,0
390 GOTO 180:REM........................BEWEGUNG WIEDERHOLEN
400 REM.................................DATEN LOK
500 DATA 0,24,0,0,19,255,0,51,255,0,227,3,1,243,3,1
510 DATA 243,3,1,243,3,29,243,3,29
520 DATA 243,3,29,243,3,255,255,255,255,255
530 DATA 255,255,255,255,241,255,143,238,255
540 DATA 119,223,126,251,191,189,252,63,153
550 DATA 252,63,129,252,31,0,248,14,0,112
555 REM.................................DATEN WAGEN
560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
570 DATA 0,0,0,0,0,0,0,0,0,0,0,0,63,255
580 DATA 255,63,255,255,63,255,255,60,127,31
590 DATA 59,190,239,55,221,247,239,235,251,15
600 DATA 227,248,15,227,248,7,193,240,3,128,224
```



```
10 REM.................................BEISPIEL 2.5
20 PRINT"□"
40 V=53248:REM...........................VIDEOCHIP
100 GOSUB 1000: REM.....................TUNNEL
120 POKE 2042,13:POKE 2043,14:POKE 2044,14:REM BLOCK NR.
140 FOR I=0TO125:READQ:POKE832+I,Q:NEXTI:REM DATEN EINLESEN
160 POKEV+41,0:POKEV+42,7:POKEV+43,2:REM.FARBE WAGEN
200 POKE V+21,4:ZW=0: REM................LOK
210 FORX=255TO  0STEP-1:REM..............BEWEGUNG
212 IF PEEK(V+31)=12 THEN POKEV+27,4:REM.TUNNEL ERREICHT?
214 IF PEEK(V+31)>12ANDZW=0 THEN POKEV+27,12:ZW=1
216 IF (PEEK(V+31)>=28ANDX<=23) THEN POKEV+27,28
220 POKEV+4,X:POKEV+5,160
225 REM.................................ERSTER WAGEN
230 IF X<=230ANDPEEK(V+21)<28 THEN POKEV+21,12
240 IF X<=230 THEN POKEV+6,X+25:POKEV+7,160
245 REM.................................ZWEITER WAGEN
250 IF X<=205ANDPEEK(V+21)<28 THEN POKEV+21,28
260 IF X<=205 THEN POKEV+8,X+50:POKEV+9,160
270 NEXTX
280 POKEV+6,0:POKEV+7,160:POKEV+8,0:POKEV+9,160
370 REM....................AUSGANGSZUSTAND WIEDER HERSTELLEN
380 POKEV+21,0:POKEV+27,0:POKEV+31,0
390 GOTO 200:REM........................BEWEGUNG WIEDERHOLEN
400 REM.................................DATEN LOK
500 DATA 0,24,0,0,19,255,0,51,255,0,227,3,1,243,3,1
510 DATA 243,3,1,243,3,29,243,3,29
520 DATA 243,3,29,243,3,255,255,255,255,255
530 DATA 255,255,255,255,241,255,143,238,255
540 DATA 119,223,126,251,191,189,252,63,153
550 DATA 252,63,129,252,31,0,248,14,0,112
```

```
555 REM...................................DATEN WAGEN
560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
570 DATA 0,0,0,0,0,0,0,0,0,0,0,0,63,255
580 DATA 255,63,255,255,63,255,255,60,127,31
590 DATA 59,190,239,55,221,247,239,235,251,15
600 DATA 227,248,15,227,248,7,193,240,3,128,224,0,0
999 REM...................................UNTERPROGRAMM TUNNEL
1000 FOR I=1TO12: PRINT: NEXTI: REM .....LEERZEILEN
1010 PRINT CHR$(18);CHR$(169);CHR$(32);CHR$(32);CHR$(32);CHR$(32);
1015 PRINT CHR$(32);CHR$(146);CHR$(169);CHR$(163);CHR$(109)
1020 PRINT CHR$(18);CHR$(32);CHR$(32);CHR$(32);CHR$(32);CHR$(32);
1030 PRINT CHR$(32);CHR$(146);CHR$(32);CHR$(32);CHR$(32);CHR$(180)
1040 PRINT CHR$(18);CHR$(32);CHR$(32);CHR$(32);CHR$(32);CHR$(32);
1050 PRINT CHR$(32);CHR$(146);CHR$(32);CHR$(32);CHR$(32);CHR$(180)
1060 PRINTCHR$(18);CHR$(164);CHR$(164);CHR$(164);CHR$(164);
1070 PRINT CHR$(164);CHR$(164);CHR$(146);
1080 FOR I=1TO30:PRINTCHR$(210);:NEXT I
1090 RETURN

READY.
```

2.5. Zusammenfassung

Sprites sind vom Benutzer frei definierbare Figuren, deren Größe durch ein Raster von 24 x 21 Bildschirmpunkten begrenzt ist.

Bis zu 8 Sprite-Figuren können gleichzeitig auf dem Bildschirm erscheinen.

Ein Sprite wird durch Angabe der Bildschirm-Koordinaten an eine beliebige Bildschirmposition gesetzt. Bewegungen von Sprites sind einfach durch Verändern der Koordinaten möglich.

Ein Sprite kann auf verschiedene Weise vergrößert werden: in Richtung der X-Achse (breit), in Richtung der Y-Achse (schmal), in beiden Richtungen (linear vergrößert).

Das Entwerfen eines Sprites kann mit Hilfe des beiliegenden Entwurfsblattes erleichtert werden. (Ein Sprite kann auch direkt am Bildschirm entworfen werden, wobei die notwendigen Berechnungen dem Computer übertragen werden. Ein Programm dazu findet sich in Kapitel 5).

<u>Wichtige Speicher-Adressen für die Erstellung von Sprites:</u>

Der Video-Chip mit der Anfangsadresse 53248 kontrolliert die Sprites-Darstellung.

Die Adresse 53248+21 enthält die Code-Zahlen der Sprites, die auf dem Bildschirm erscheinen.

Die Adressen 53248 bis 53248+16 kontrollieren die Koordinaten der Sprites.

Die Adressen 53248+23 und 53248+29 sind für das Vergrößern eines Sprites zuständig.

Die Adressen 53248+39 bis 53248+46 bestimmen die Spritefarbe, und unter den Adressen 2040 bis 2047 ist angegeben, wo die Daten für die Sprites abgelegt sind.

3. Die hochauflösende Grafik

Die Möglichkeiten der hochauflösenden Grafik werden im Handbuch leider überhaupt nicht erklärt. Dabei bieten sich gerade auf diesem Gebiet sehr interessante Anwendungsbereiche. Eine Einführung dazu - mit Beispielen - soll das folgende Kapitel liefern.

Gleich zu Beginn möchten wir aber darauf hinweisen, daß Sie bei Verwendung der hochauflösenden Grafik zum Teil erhebliche Geduld aufbringen müssen. Das liegt daran, daß der Commodore 64 keine eingebauten Maschinensprache-Routinen bzw. keine speziellen BASIC-Befehle z.B. für das Zeichnen von Geraden oder Kreisen besitzt. Ja, sogar zur Darstellung eines einzelnen Bildschirm-Punktes benötigt man ein kleines Programm! Darstellungen in der hochauflösenden Grafik bauen sich also recht langsam auf. Wir haben uns jedoch bemüht, die in diesem Kapitel vorgestellten Beispiele so auszuwählen, daß Ihre Geduld nicht auf eine allzu harte Probe gestellt wird.

3.1. Ein- und Ausschalten des hochauflösenden Grafik-Bildschirms

Wie bei den Sprites wird in der hochauflösenden Grafik der Bildschirm in 64000 Punkte aufgeteilt: 200 Zeilen zu je 320 Punkten. Im Unterschied zu den Sprites steht aber jetzt der gesamte Bildschirmbereich gleichzeitig zur Verfügung.

Die Aufgaben, um z.B. eine Funktion in der hohen Auflösung darzustellen, übernimmt wieder - wie bei den Sprites - die Speichergruppe des Video-Chips. Der erste Speicher hat, wie schon erwähnt, die Adresse 53248 (im folgenden abgekürzt durch: V).

Um den Computer zu veranlassen, von der normalen (Text)-Darstellung in die hohe Auflösung umzuschalten, muß zunächst der Wert 59 in den Speicher V+17 geschrieben werden (POKE V+17,59), sowie der Wert 24 in den Speicher V+24 (POKE V+24,24). Damit ist die hochauflösende Grafik "eingeschaltet". Auf dem Bildschirm erscheint daraufhin ein "undefiniertes" Grafikmuster.

Auch bei dieser Form der Grafik ist die Angabe einer Farbe erforderlich. Dafür ist wie immer eine besondere Speichergruppe zuständig. Bei der hochauflösenden Grafik gibt es aber nun nicht noch eine weitere Speichergruppe für die Farbe. Vielmehr wird diese Aufgabe von dem schon bekannten Bildschirmspeicher übernommen (zur Erinnerung: das sind die Speicherplätze von 1024 bis 2023). In diese Speicherstellen werden nun gleichzeitig die Code-Zahlen für Hintergrund- und Punktfarbe geschrieben. Der zu POKEnde Wert berechnet sich folgendermaßen: Die Code-Zahl für die Punktfarbe wird mit 16 multipliziert und zu dem Ergebnis die Code-Zahl für die Hintergrundfarbe addiert. Haben Sie sich für den gesamten Bildschirm für eine Hintergrund- (HF) und eine Punktfarbe (PF) entschieden, schreiben Sie folgende Anweisung in Ihr Programm:

```
FOR I = 1024 TO 2023: POKE I,HF+16*PF: NEXT I
```

Da jetzt aber Farb-Codes im Bildschirmspeicher stehen, können nicht gleichzeitig auch noch Zeichen-Codes (also etwa Buchstaben) benutzt werden. Das Mischen von hochauflösender Grafik und Text ist also - von BASIC aus - nicht möglich! (Man kann allerdings selbst-definierte Zeichen in der Grafik verwenden. Von dieser Möglichkeit wurde z.B. in dem Programm zum Zeichnen von Funktionen - in Kapitel 5 - Gebrauch gemacht.)

Die Daten für die hochauflösende Grafik werden unter den Adressen 8192 bis 16383 gespeichert (vgl. Kap. 7). Nach dem Umschalten auf die Grafik-Darstellung sollten zunächst die bisherigen grafischen Daten gelöscht werden, damit nicht die zuletzt erstellte Grafik mit der neuen vermischt wird. Um alte Daten zu löschen, schreibt man einfach den Wert 0 in die erwähnten Speicherplätze:

```
FOR I = 8192 TO 16383: POKE I,0: NEXT I
```

Damit hat man den Bildschirm gelöscht, auf dem nun die gewünschte Grafik erstellt werden kann, indem die entsprechenden Daten in die jeweiligen Speicherzellen eingetragen werden. Dazu muß man die Aufteilung dieses Speicherbereiches kennen:

Speicher-Aufteilung der hochauflösenden Grafik

Jeder normale Bildschirmpunkt wird aufgeteilt in 8 Zeilen, und für jede dieser Zeilen ist ein Speicherplatz zuständig. Als Beispiel betrachten wir die Zeile, die zu dem Speicherplatz 8192 (ganz oben links) gehört. Diese Zeile ist selbst wieder unterteilt in 8 einzelne Kästchen:

	128	64	32	16	8	4	2	1
8192								

Ebenso wie bei den Sprites benötigt der Computer wieder die verschlüsselte Angabe darüber, welches Kästchen in welcher Zeile ausgefüllt werden soll. Die Nummer der Zeile haben wir ja bereits: 8192. Über die Kästchen einer Zeile schreibt man - wie bei den Sprites - die Zahlen 128, 64, ... 1, füllt die gewünschten Kästchen aus, und addiert die Zahlen, die über den ausgefüllten Kästchen einer Zeile stehen. Das Ergebnis ist der Wert, der in den Speicherplatz in der entsprechenden Zeile geschrieben werden muß.

Beispiel:

	128	64	32	16	8	4	2	1
8192			X		X		X	

ergibt

```
POKE 8192,32+8+2 bzw.
POKE 8192,42
```

Nun kann man die entsprechende Methode für Sprites ja gerade noch anwenden, da Sprites eine genau festgelegte, begrenzte Größe haben. Möchte man aber in der hochauflösenden Grafik z.B. einen Kreis oder eine Sinus-Kurve auf den Bildschirm bringen, so ist dieses Verfahren zur Berechnung der einzelnen Bildschirmpunkte einfach nicht mehr möglich.

Die Lösung dieses Problems: man läßt den Computer selbst ausrechnen, welche Werte in welche Speicher geschrieben werden müssen! - Dazu dient das folgende Unterprogramm, das auch in selbsterstellten Programmen sehr gut benutzt werden kann:

```
59999 REM..............................BEISPIEL 3.1
60000 REM..............................PUNKT SETZEN
60010 XK=8*INT(X/8)
60015 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60020 EX=2↑(7-INT((X/8-INT(X/8))*8))
60025 S=8192+XK+YK
60030 POKE S,PEEK(S) OR EX
60035 RETURN
```

In diesem Programm geschieht folgendes:

Zeile 60010: Die Werte der X-Koordinaten können bei der hochauflösenden Grafik zwischen 0 und 319 liegen. Da aber ein Speicherplatz jeweils 8 X-Koordinatenpunkte kontrolliert, wird der X-Wert aus dem Hauptprogramm so umgerechnet, daß er genau auf das erste Kästchen eines Speicherplatzes zeigt.

Zeile 60015: Hier wird der Y-Wert aus dem Hauptprogramm so umgerechnet, daß er genau auf den ersten Speicher eines normalen Bildschirmkästchens weist (320 * INT(Y/8)). Der zweite Teil der Formel, ((Y/8) - INT(Y/8)) * 8, ergibt den ganzzahligen Rest der Division Y/8 und setzt die Y-Koordinate in die richtige Reihe des "normalen" Bildschirmkästchens.

Zeile 60020: Hier wird errechnet, welcher Wert in den Speicher S geschrieben werden muß. Dies hängt davon ab, welcher Punkt ausgefüllt werden soll: 128 = 2↑7, 64 = 2↑6 usw. Auch hier wird wieder der ganzzahlige Rest aus einer Division durch 8 ausgerechnet.

Zeile 60025: Hier wird der richtige Speicherplatz ausgerechnet. Die Grafikdatenspeicher beginnen ja bei 8192; dazu werden die Werte für XK und YK addiert.

Zeile 60030: PEEK(S) OR EX bedeutet, daß in den Speicher S der ausgerechnete Wert geschrieben wird (falls dieser Speicher vorher noch nicht durchlaufen wurde), oder der alte Wert + EX (falls schon vorher ein Punkt gesetzt war, jetzt aber ein neuer dazukommt), oder 255 (falls schon alle Punkte gesetzt sind, in dem Falle = PEEK(S)).

3.2. Standard-Grafik-Routinen. Beispielprogramme zum Zeichnen von Punkten, Kreisen, Geraden.

Da, wie erwähnt, im Standard-BASIC des Commodore 64 leider keine spezifischen Anweisungen zur Grafik-Gestaltung enthalten sind, ist es außerordentlich aufwendig, auch nur einen einzigen Bildpunkt in der hochauflösenden Grafik zu setzen. Die Programme aus diesem Kapitel sollen diesem Mangel ein wenig abhelfen. Sie lassen sich als Unterprogramme in eigenen Programmen benutzen. Da es sich um reine BASIC-Programme handelt, können sie von der Geschwindigkeit her natürlich nicht mit entsprechenden Hilfs-Programmen konkurrieren, die in Maschinensprache geschrieben sind.

Die Programmzeilen wurden so numeriert, daß diese Unterprogramme gleichzeitig im Arbeitsspeicher stehen können.

Die jeweils benötigten Werte werden im Hauptprogramm bestimmten, dafür reservierten Variablen zugewiesen. Diese Variablen werden dann im jeweiligen Unterprogramm benutzt. Die Programme werden so erheblich übersichtlicher.

P u n k t - D a r s t e l l u n g

Um in der hochauflösenden Grafik-Darstellung einen Punkt an eine bestimmte Bildschirmposition zu setzen, werden die beiden dazu notwendigen Koordinatenangaben den beiden Variablen X und Y zugewiesen. Anschließend erfolgt ein Aufruf des betreffenden Unterprogramms, das den Punkt auf den Bildschirm setzt:

```
10 REM............................GRAFIK-DARSTELLUNG VORBEREITEN
20 V=53248
30 POKE V+17,59: POKE V+24,24: REM.......GRAFIK EINSCHALTEN
40 FOR I=1024TO2023: POKE I,14: NEXTI: REM.......GRUNDFARBE
50 FOR I=8192TO16383: POKE I,0: NEXTI: REM GRAFIK-SPEICHER LOESCHEN
60 :
100 REM.........................ZUWEISUNG ZWEIER KOORDINATENWERTE
110 X=160
120 Y=100
130 GOSUB 60000: REM.....................PUNKT X,Y SETZEN
140 END
150 :
60000 REM...............................PUNKT SETZEN
60010 XK=8*INT(X/8)
60015 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60020 EX=2↑(7-INT((X/8-INT(X/8))*8))
60025 S=8192+XK+YK
60030 POKE S,PEEK(S) OR EX
60035 RETURN
```

In den Zeilen 60010 und 60015 werden die Grafik-Speicherplätze für die X- bzw. Y-Koordinate des Punktes berechnet. In der folgenden Zeile (60020) wird der Wert berechnet, der schließlich in den aktuellen Grafikspeicher S (60025) geschrieben wird (60030) und die Darstellung des Punktes auf dem Bildschirm bewirkt.

Zum Löschen eines Punktes dient ein ganz ähnlich aufgebautes Unterprogramm:

```
60130 REM...............................BEISPIEL 3.2
60150 REM...............................PUNKT LOESCHEN
60160 XK=8*INT(X/8)
60165 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60170 S=8192+XK+YK
60175 POKE S,0
60180 RETURN
```

Im Unterschied zu dem Programm, das einen Punkt setzt, wird hier in die genauso berechnete Speicherzelle S der Wert 0 geschrieben, wodurch der Punkt auf dem Bildschirm gelöscht wird (Zeile 60175).

Da sich alle möglichen Grafik-Darstellungen aus solchen Bildschirmpunkten zusammensetzen, lassen sich mit Hilfe dieser beiden Unterprogramme im Prinzip auch alle möglichen Bildschirm-Grafiken erstellen - sofern keine schnell bewegten Bilder verlangt werden.

Zeichnen eines Kreises

Lassen wir doch z.B. unseren Punkt einmal auf einer Kreisbahn herumwandern, also einen Kreis zeichnen! Dazu machen wir einen kleinen Abstecher in die analytische Geometrie - die übrigens ein heißer Tip ist, wenn Sie näher in die Grafik-Programmierung einsteigen wollen!

Wenn man einen Kreis so darstellt, daß sein Mittelpunkt im Ursprung (Nullpunkt) eines Koordinaten-Systems liegt, so läßt sich der funktionale Zusammenhang zwischen den beiden Koordinaten X und Y und dem Radius des Kreises durch die folgende Mittelpunktsgleichung des Kreises beschreiben:

```
R↑2 = X↑2 + Y↑2
```

Löst man diese Gleichung nach X und Y auf, erhält man die Parameterdarstellung des Kreises:

```
X = R * COS(PHI),    Y = R * SIN(PHI),
```

wobei der Wert des Winkels PHI zwischen 0 und 360 Grad liegt. (Abb. 3.1).

Soll nun der Mittelpunkt eines Kreises nicht im Ursprung des Koordinaten-Systems liegen, so ergibt sich eine Verschiebung sämtlicher Kreispunkte um den Wert des Kreismittelpunktes, dessen Koordinaten mit XM und YM bezeichnet sein sollen. Damit können wir die Normalgleichung des Kreises oder die allgemeine Kreisgleichung formulieren:

```
R↑2 = (X - XM)↑2 + (Y - YM)↑2
```

Die entsprechende <u>Parameterform</u> lautet demnach (Abb. 3.2):

X = XM + R * COS(PHI),   Y = YM + R * SIN(PHI)

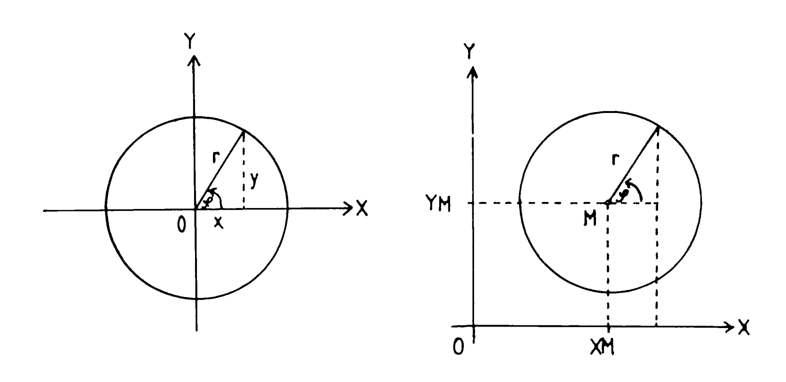


Abb. 3.1   Abb. 3.2

Damit das Kreis-Programm läuft, muß natürlich außer dem Unterprogramm bei Zeile 60000 auch noch der Programmteil zur Vorbereitung der Grafik-Darstellung im Computer stehen (Zeile 200 bis 280).

Durch Verändern der Schrittweite in Zeile 300 (STEP..) kann die Kreislinie dichter (kleinere Schrittweite) oder mit weniger Punkten (größere Schrittweite) gezeichnet werden. Probieren Sie es doch einfach einmal aus!

Damit keine falschen Werte eingegeben werden können, haben wir die Programmzeilen 1000 bis 1030 eingebaut. Außerdem können Sie den fertig gezeichneten Kreis durch Drücken der 'L'-Taste wieder löschen (Zeile 410) und mit einer anderen Taste auf den Textmodus umschalten (Zeile 400 bzw. 420).

```
10 REM..................................BEISPIEL 3.3
20 REM.................................DARSTELLUNG EINES KREISES
30 REM
100 INPUT"X-KOORDINATE MITTELPKT";X1
105 PRINT
110 IF(X1<0ORX1>320)THEN GOSUB 1000 :GOTO 100
120 INPUT"Y-KOORDINATE MITTELPKT";Y1
125 PRINT
130 IF(Y1<0ORY1>200)THEN GOSUB 1000 :GOTO 120
135 PRINT
140 INPUT"RADIUS";R
145 PRINT
150 IF(X1-R<0ORX1+R>320ORY1-R<0ORY1+R>200)THENGOSUB1000:GOTO140
200 V=53248:REM..........................VIDEOCHIP
240 POKEV+17,59:POKEV+24,24:REM..........GRAPHIKMODUS
260 FORI=1024TO2023:POKEI,14:NEXTI:REM...FARBE
280 FORI=8192TO16383:POKEI,0:NEXTI:REM...SPEICHER LOESCHEN
300 FORI=-πTOπSTEP(3/R)
320 X=X1+R*COS(I):REM....................KREISGLEICHUNG
340 Y=Y1+R*SIN(I)
360 GOSUB 60000
380 NEXTI
400 GETZ$:IFZ$=""THENGOTO 400
405 REM..................................KREIS LOESCHEN
410 IFZ$="L"THEN GOSUB 60150:GOTO 400
420 POKE53248+17,155:POKE53248+24,21:PRINT"♥"
430 REM
520 END
550 REM
1000 PRINT"UNGUELTIGE EINGABE !"
1010 PRINT"NOCHMAL WIEDERHOLEN"
1020 PRINT
1030 RETURN
```

```
59998 REM................................UNTERPROGRAMME
60000 REM................................ZEICHNEN
60010 XK=8*INT(X/8)
60015 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60020 EX=2↑(7-INT((X/8-INT(X/8))*8))
60025 S=8192+YK+XK
60030 POKES,PEEK(S) OR EX
60035 RETURN
60150 REM................................LOESCHEN
60160 FORI=-πTOπSTEP(3/R)
60170 X=X1+R*COS(I)
60180 Y=Y1+R*SIN(I)
60190 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60200 XK=8*INT(X/8)
60210 S=8192+YK+XK
60220 POKES,0
60230 NEXTI
60240 RETURN

READY.
```

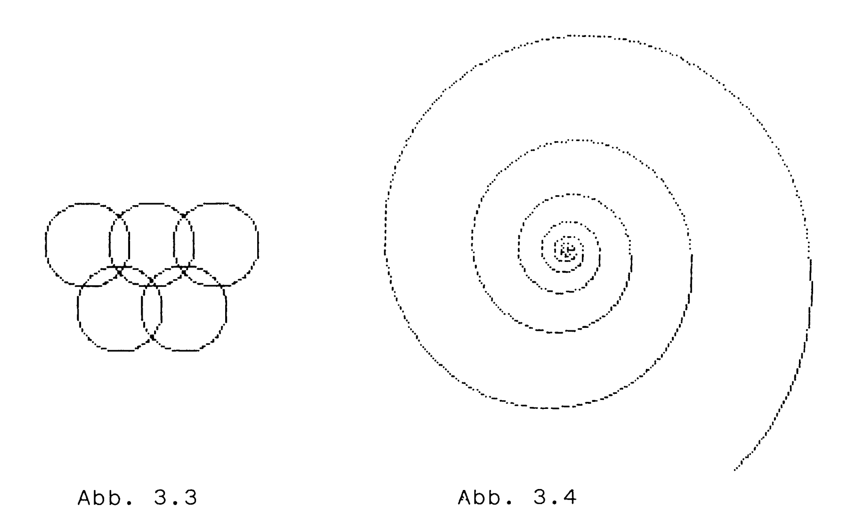


Abb. 3.3 Abb. 3.4

Abbildung 3.3 ist mit dem Kreis-Programm 3.3 erstellt worden.

Das Spiralen-Programm (Beispiel 3.4 sowie Abb. 3.4) ist durch eine nur kleine Abänderung des Kreis-Programms entstanden: die Spirale ergibt sich dadurch, daß der Radius fortwährend vergrößert wird.

```
20 REM................................SPIRALE
30 REM
80 PRINT
100 INPUT"X-KOORDINATE MITTELPKT";X1
105 PRINT
110 IF (X1<0ORX1>320) THEN PRINT"UNGUELTIGE KOORDINATE": GOTO 80
115 PRINT
120 INPUT"Y-KOORDINATE MITTELPKT";Y1
125 PRINT
130 IF (Y1<0ORY1>200) THEN PRINT"UNGUELTIGE KOORDINATE": GOTO 115
135 PRINT
140 INPUT"RADIUS";R
145 PRINT
150 IF (X1-R<0ORX1+R>320) THEN PRINT"RADIUS ZU GROSS": GOTO 140
160 IF (Y1-R<0ORY1+R>200) THEN PRINT"RADIUS ZU GROSS": GOTO 140
180 REM...............................GRAPHIK EINSCHALTEN
200 V=53248
220 POKE V+17,59: POKE V+24,24
240 FOR I=1024TO2023: POKE I,14: NEXTI
260 FOR I=8192TO16383: POKE I,0: NEXTI
280 REM...............................SPIRALENFUNKTION
300 FOR I=-πTO(π-π/360) STEP(2/R)
320 X=X1+R*COS(I)
340 Y=Y1+R*SIN(I)
350 IF (Y<0ORY>200ORX<0ORX>320) THEN GOTO 380
360 GOSUB 60000
370 R=R+.15: REM........................VERGROESSERUNG DES RADIUS
380 NEXTI
440 GOTO 300: REM.......................WIEDERHOLUNG
60000 XK=8*INT(X/8)
60020 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60040 EX=2↑(7-INT((X/8-INT(X/8))*8))
60060 S=8192+YK+XK
60080 POKES,PEEK(S) OR EX
60100 RETURN

READY.
```

Zeichnen einer Geraden

Mit Hilfe des beschriebenen Unterprogramms zum Setzen eines Punktes in der hochauflösenden Grafik können Sie nun beliebige Darstellungen - beispielsweise mathematische Funktionen - erzeugen. Dabei wird dieses Unterprogramm immer wieder aufgerufen, sodaß die einzelnen Punkte, z.B. entsprechend einer Funktionsgleichung, der Reihe nach auf den Bildschirm gesetzt werden.

Die Gleichung einer Geraden wird durch die Funktion

$$y = a x + b$$

angegeben. Dabei bedeutet a die Steigung der Geraden und b den y-Achsen-Abschnitt (Abb. 3.5).

Wollen Sie nun eine bestimmte Gerade, deren Steigung und y-Achsen-Abschnitt bekannt ist, auf dem Bildschirm sehen, können Sie - bis auf eine wichtige Änderung - diese Geradengleichung in Ihr Programm übernehmen. Diese Änderung ist deshalb notwendig, weil das Koordinaten-System der Bildschirmpunkte nicht mit dem üblichen (kartesischen) Koordinaten-System übereinstimmt, denn es gibt keine negativen Koordinaten für einen Bildschirmpunkt.

Da der Nullpunkt der Bildschirmkoordinaten in der linken oberen Ecke liegt (Abb.3.6), muß die allgemeine Geradengleichung für den Bildschirm demnach lauten:

$$y = 200 - (a x + b)$$

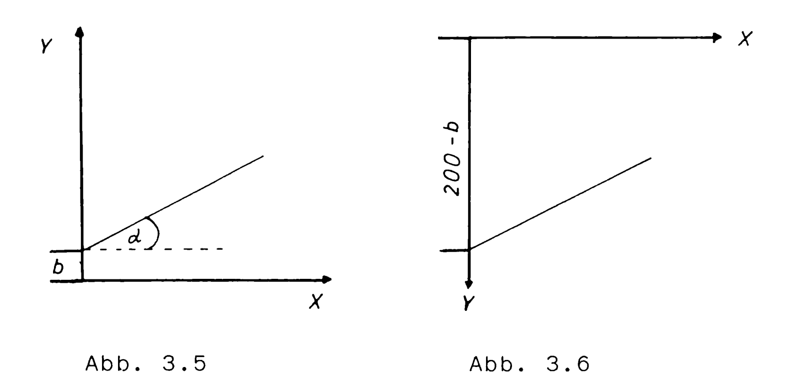


Abb. 3.5 Abb. 3.6

Mit konkreten Werten könnte Ihr Geraden-Programm also z.B. folgendermaßen aussehen:

```
10 REM...........................GRAFIK-DARSTELLUNG VORBEREITEN
20 V=53248
30 POKE V+17,59: POKE V+24,24:REM...........GRAFIK EINSCHALTEN
40 FOR I=1024TO2023: POKE I,14: NEXT I:REM................FARBE
50 FOR I=8192TO16383: POKE I,0: NEXT I:REM.GRAFIKSPEICHER LOESCHEN
100 REM........................................GERADE ZEICHNEN
110 FOR X=0TO319
120 Y=200-0.5*X-5
130 GOSUB60000:REM..............................PUNKT SETZEN
140 NEXT X
```

Wir empfehlen Ihnen jedoch, vor dem Aufruf des Unterprogrammes überprüfen zu lassen, ob die Werte für X und Y noch im Bereich des Bildschirmes liegen. Sonst kann es in ungünstigen Fällen passieren, daß Ihr Programm zerstört wird! Eine entsprechende Anweisung sollte also nie fehlen:

```
125 IF (X<O OR X>319 OR Y<O OR Y>199)THEN GOTO 140
```

Nun kennen Sie aber nicht in jedem Falle die Steigung und den Y-Achsenabschnitt einer Geraden. Dann bietet sich eine andere Form der Geradengleichung an, bei der die Gerade definiert wird durch zwei feste Punkte, die sie durchläuft. Haben Sie also zwei Punkte mit den Koordinaten X1 und Y1 bzw. X2 und Y2, und wollen diese beiden Punkte durch eine Gerade verbinden, so lautet die entsprechende Funktionsgleichung:

$$(Y - Y1) / (Y2 - Y1) = (X - X1) / (X2 - X1)$$

bzw.

$$Y = ((X - X1) (Y2 - Y1)) / (X2 - X1) + Y1$$

In dieser Form können Sie diese Geradengleichung in ein Programm übernehmen.

In Beispiel 3.4 finden Sie ein Programm zur Darstellung beliebiger Geraden, definiert durch zwei Punkte. Die Programmteile zur Grafik-Vorbereitung und das Unterprogramm zum Punkt setzen stimmen mit den bereits bekannten Programmen überein.

In diesem Programm wurde auch der Sonderfall berücksichtigt, der mit dieser Geradengleichung nicht dargestellt werden kann: eine senkrechte Linie, d.h. X1 = X2.

In Beispiel 3.5 wird das Programm zum Geraden-Zeichnen dazu verwendet, um einen Würfel darzustellen.

```
10 REM..................................BEISPIEL3.4
20 REM..................................DARSTELLUNG EINER GERADEN
30 REM
100 INPUT"X-KOORDINATE 1.PUNKT";X1
120 INPUT"Y-KOORDINATE 1.PUNKT";Y1
130 PRINT
140 INPUT"X-KOORDINATE 2.PUNKT";X2
160 INPUT"Y-KOORDINATE 2.PUNKT";Y2
170 PRINT
220 V=53248:REM.........................VIDEOCHIP
260 POKEV+17,59:POKEV+24,24:REM.........GRAPHIKMODUS
280 FORI=1024TO2023:POKEI,14:NEXTI:REM...FARBE
300 FORI=8192TO16383:POKEI,0:NEXTI:REM...LOESCHEN
```

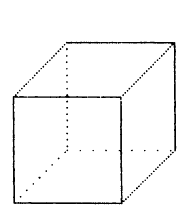
```
310 REM.................................KOORDINATEN ZEICHNEN
320 FORX=3TO319:Y=198
340 GOSUB60000
360 NEXTX
380 FORX=3TO319STEP20:Y=199
400 GOSUB60000
420 NEXTX
440 FORX=3TO319STEP20:Y=200
460 GOSUB60000
480 NEXTX
500 FORY=0TO199:X=3
520 GOSUB60000
540 NEXTY
560 FORY=0TO199STEP20:X=2
580 GOSUB60000
600 NEXTY
620 FORY=0TO199STEP20:X=1
640 GOSUB60000
660 NEXTY
680 IF X1=X2 THEN GOTO 790
700 FORX=X1TOX2
710 REM.................................GERADENGLEICHUNG
720 Y=((X-X1)*(Y2-Y1)/(X2-X1))+Y1
740 GOSUB60000
780 NEXTX:GOTO820
790 FOR Y=Y1TOY2
800 X=X1
810 GOSUB 60000
815 NEXT Y
820 GETZ$
840 REM.................................AUF TEXT SCHALTEN
860 IFZ$=""GOTO820
880 POKEV+17,155:POKEV+24,21:PRINT"J"
900 END
50999 REM.................................ZEICHNEN
60000 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60020 XK=8*INT(X/8)
60040 EX=2↑(7-INT((X/8-INT(X/8))*8))
60060 S=8192+YK+XK
60080 POKES,PEEK(S) OR EX
60100 RETURN

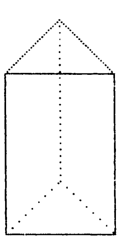
READY.
```

```
10 V=53248: REM.........................GRAFIK EINSCHALTEN
20 POKE V+17,56: POKE V+22,15: POKE V+24,24
30 FOR I=1024TO2023: POKE I,1: NEXT I
40 FOR I=8192TO16383: POKE I,0: NEXT I
45 REM..................................ZEICHNEN DES WUERFELS
50 FOR X=50TO100
55 Y=70: GOSUB60000
60 Y=120:GOSUB60000
70 NEXT X
80 FOR Y=70TO120
85 X=50: GOSUB60000
90 X=100: GOSUB60000
100 NEXT Y
110 FOR X=100TO125
115 Y=120-(X-100):GOSUB60000
120 Y=70-(X-100): GOSUB60000
130 NEXT X
140 FOR X=50TO75
145 Y=70-(X-50): GOSUB60000
150 NEXT X
160 FOR X=75TO125
165 Y=45: GOSUB60000
170 NEXT X
180 FOR Y=45TO95
185 X=125: GOSUB60000
190 NEXT Y
200 FOR X=50TO75 STEP3
205 Y=120-(X-50):GOSUB60000
210 NEXT X
220 FOR X=75TO125 STEP6
225 Y=95: GOSUB60000
230 NEXT X
240 FOR Y=45TO95 STEP3
245 X=75: GOSUB60000
250 NEXT Y
```

```
260 REM.................................ZEICHNEN DES PRISMAS
270 FOR X=200TO250
275 Y=120: GOSUB60000
280 Y=45: GOSUB60000
290 NEXT X
300 FOR Y=45TO120
305 X=200: GOSUB60000
310 X=250: GOSUB60000
320 NEXT Y
330 FOR X=200TO225
335 Y=45+(200-X): GOSUB60000
340 NEXT X
350 FOR X=225TO250
355 Y=20-(225-X): GOSUB60000
360 NEXT X
370 FOR X=200TO225 STEP3
375 Y=120+(200-X):GOSUB60000
380 NEXT X
390 FOR X=225TO250 STEP3
395 Y=95-(225-X): GOSUB60000
400 NEXT X
410 FOR Y=20TO95 STEP3
415 X=225: GOSUB60000
420 NEXT Y
430 END
60000 REM........................UNTERPROGRAMM ZUM PUNKT SETZEN
60010 XK=8*INT(X/8)
60020 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60030 S=8192+XK+YK
60040 EX=2↑(7-INT((X/8-INT(X/8))*8))
60050 POKE S,PEEK(S)OREX
60060 RETURN

READY.
```





3. Z u s a m m e n f a s s u n g

In der hochauflösenden Grafik wird der Bildschirm in 320 x 200 Einzelpunkte aufgeteilt. Jeder einzelne dieser 64000 Bildschirmpunkte kann direkt angesprochen werden und gesetzt oder nicht gesetzt sein. Die Daten für die hochauflösende Grafik werden unter den Adressen 8192 bis 16383 gespeichert. Die wichtigsten Anweisungen für die hochauflösende Grafik sind:

Grafik einschalten:

Zum Umschalten vom Text- in den Grafikmodus sind folgende Anweisungen erforderlich:

```
POKE V+17,59: POKE V+24,24
```

wobei V = 53248 ist.

Farbe wählen:

Durch die Anweisungen

```
FOR I = 1024 TO 2023: POKE I,HF + 16 * PF: NEXT I
```

wird eine Punktfarbe (PF) und eine Hintergrundfarbe (HF) gewählt.

Grafik löschen:

Mit den Anweisungen

```
FOR I = 8192 TO 16383: POKE I,0: NEXT I
```

wird der Grafik-Speicher gelöscht.

4. Etwas Theorie

ROM- und RAM-Speicher

In den vorangegangenen Kapiteln und den vorgestellten Beispielen haben wir immer wieder speziellen Speicherplätzen des Computers bestimmte Werte zugewiesen (mit POKE). Was passiert dabei eigentlich?

Wir haben schon erklärt, daß viele Speicher des Computers für eine ganz bestimmte Aufgabe zuständig sind. Diese lassen sich zu mehreren Gruppen zusammenfassen. Grundsätzlich besteht der gesamte Speicher des Computers aus zwei Bereichen:

1. Der ROM-Bereich (Read Only Memory), ein Speicherbereich, in dem Daten fest gespeichert sind und auch nach dem Ausschalten des Computers nicht gelöscht werden. Die Daten aus diesen Speichern können nur gelesen, jedoch nicht vom Benutzer verändert werden. In solchen Speichern befindet sich z.B. der Interpreter, der die Anweisungen eines BASIC-Programms in die Maschinensprache des Computers übersetzt. Könnte der Benutzer in diesen "geschützten" Speicherbereichen etwas ändern, wäre es leicht möglich, daß der Computer plötzlich kein BASIC mehr verstünde und deshalb nicht mehr in der Lage wäre, irgendein Programm auszuführen.

2. Der RAM-Bereich (Random Access Memory) ist ein Schreib- und Lesespeicher. In diesem Speicherbereich stehen die Anwender-Programme und -Daten. Außerdem befinden sich hier der Bildschirm- und der Farbspeicherbereich sowie andere besondere Speicherplätze, die eine ganz bestimmte Aufgabe haben. Mit Hilfe des PEEK-Befehls kann man den Wert einer Speicherzelle des RAM-Bereichs lesen, mit Hilfe des POKE-Befehls kann man den Wert einer Speicherzelle verändern, indem man einen neuen Wert hineinschreibt.

Bit und Byte

Die Speicherplätze sind aber nicht die kleinsten Einheiten im Computer. Jeder Speicher ist noch einmal unterteilt in 8 Einheiten (bits). Diese Bits haben eine festgelegte Reihenfolge (Bit 0 bis Bit 7). Die 8 Bits zusammen bilden ein Byte. Einen Speicherplatz des Computers kann man sich demnach folgendermaßen vorstellen:

Speicheradresse (z.B. 53248)

7	6	5	4	3	2	1	0

Bit-Nr.

Die einzelnen Bits eines Speichers können nun "gesetzt" oder "nicht gesetzt", "ein-" oder "ausgeschaltet" werden. Die verschiedenen Kombinationen von gesetzten und nicht-gesetzten Bits einer Speicherzelle geben dem Computer an, wie er im konkreten Fall die Aufgabe dieses Speichers auszuführen hat. Wie kommt es, daß wir diesen Bits bisher noch nicht begegnet sind? Wenn wir z.B. einen POKE-Befehl gegeben haben, haben wir dem Computer keineswegs gesagt: "In Speicher Nr. 53248 setze Bit 0, setze nicht Bit 1, setze Bit 3, ... usw." Wo ist da der Zusammenhang? Mit den POKE-Befehlen haben wir Dezimalzahlen in die Speicher geschrieben. Diese Werte werden vom Computer in Binär- oder Dualzahlen umgerechnet, um zu wissen, welche Bits "ein-" und welche "ausgeschaltet" werden müssen. Dabei wird für "eingeschaltet" eine 1 und für "ausgeschaltet" eine 0 gesetzt. Da der Computer nur zwischen diesen beiden Zuständen unterscheiden kann, benutzt man zur Beschreibung dieser Vorgänge also das binäre (oder duale) Zahlensystem.

D a s B i n ä r - S y s t e m

Das binäre (oder duale) Zahlensystem (dual bzw. binär = 2) verwendet nur die beiden Ziffern 0 und 1. Genau wie im Dezimalsystem bestimmt auch im Dualsystem die Stelle innerhalb einer Zahl, an der eine Ziffer des Systems steht, ihren Wert (z.B. dezimal 296 = 1 * 6 + 10 * 9 + 100 * 2). Im Dezimalsystem werden die Ziffern entsprechend dem Wert ihrer Position mit Potenzen von 10 multipliziert: 10↑0 = 1, 10↑1 = 10, 10↑2 = 100 usw. Im Dualsystem werden stattdessen Potenzen von 2 verwendet: 2↑0 = 1, 2↑1 = 2, 2↑2 = 4, 2↑3 = 8 usw. Die Dualzahl 1001 wird demnach folgendermaßen (von rechts nach links) in eine Dezimalzahl umgerechnet:

```
1001 = 1 * 2↑0 + 0 * 2↑1 + 0 * 2↑2 + 1 * 2↑3
     = 1 * 1   + 0 * 2   + 0 * 4   + 1 * 8
     = 9
```

Da ein Speicherplatz des Computers aus 8 Bits besteht, die alle den Wert 0 oder 1 haben können, ist

der kleinste Wert, den ein Speicher aufnehmen kann:

```
00000000 (dual) = 0       (kein Bit gesetzt)
```

und der größte:

```
11111111 (dual) = 1 * 2↑0 + 1 * 2↑1 + 1 * 2↑2 +
                  1 * 2↑3 + 1 * 2↑4 + 1 * 2↑5 +
                  1 * 2↑6 + 1 * 2↑7
                = 255    (alle Bits gesetzt).
```

Das ist also die Erklärung dafür, daß in keinen Speicher ein Wert, der größer als 255 ist, geschrieben werden kann.

Außerdem können Sie sich nun das Sprite-Entwurfsblatt als eine Gruppe von Bits vorstellen, die die Werte 0 oder 1 annehmen. Die Einteilung jeder Zeile in drei Gruppen zu je 8 Kästchen ist also deshalb notwendig, weil jedes Datenbyte aus genau 8 Bits besteht. Beim Addieren der Werte der ausgefüllten Kästchen haben Sie also schon Dualzahlen in Dezimalzahlen umgerechnet.

Das Hexadezimal-System

Häufig wird in Programmierhandbüchern noch ein weiteres Zahlensystem verwendet, das Hexadezimalsystem. Im Commodore-Handbuch stehen z.B. in der Speicherbelegungstabelle als erste Angaben die Speicheradressen im Hexadezimalsystem (Hex-Code).

Dies ist ein Zahlen-System, das auf 16 (hexadezimal = 16) verschiedenen Zahlzeichen aufgebaut ist. Wie im Dezimal-System gibt es die Ziffern 0 bis 9. Hinzu kommen die Buchstaben A bis F, die als Zahlzeichen den (dezimalen) Werten von 10 bis 15 entsprechen. Wieder bestimmt die Stelle innerhalb einer Zahl den Wert der Ziffer, nur muß man jetzt mit Potenzen von 16 rechnen:

```
A90F = 15 * 16↑0 + 0 * 16↑1 + 9 * 16↑2 + 10 * 16↑3
       (F)                                (A)
     = 43279 (dezimal).
```

In diesem System lassen sich also mit nur 4 Stellen recht große Zahlen darstellen. Für die Dezimal-

zahlen von 0 bis 255, für die man ja im Dual-System bis zu 8 Stellen braucht, reichen hier zwei Stellen aus: 255 (dezimal) = FF (hexadezimal). Die größte mit 4 Stellen darstellbare Zahl im Hexadezimal-System (also: FFFF) ist übrigens 65535 - das entspricht 64K, der gesamten Speicherkapazität des Commodore 64. D.h. das man mit einer 4-stelligen Hexadezimal-Zahl sämtliche Adressen (Speicherplätze) des Computers benennen kann.

Die folgende Tabelle enthält die Zahlen von 0 bis 32 in dezimaler, binärer und hexadezimaler Darstellung:

Dezimal	Binär	Hexadezimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14
21	10101	15
22	10110	16
23	10111	17
24	11000	18
25	11001	19
26	11010	1A
27	11011	1B
28	11100	1C
29	11101	1D
30	11110	1E
31	11111	1F
32	100000	20

5. Beispiel-Programme

Nach der - hoffentlich nicht allzu grauen - Theorie sollen in diesem Kapitel die verschiedenen Grafik-Möglichkeiten des Commodore 64 an etwas umfangreicheren Programmen beispielhaft vorgeführt werden. Außerdem finden Sie hier die - zum Teil erweiterten - vollständigen Programme zu den ersten Kapiteln, sofern sie dort als Beispiele nur abschnittsweise angeführt wurden.

Zum besseren Verständnis ist jedem Programm eine kurze Erklärung vorangestellt. Für die etwas komplexeren Programme ist auch ein Flußdiagramm angegeben, an Hand dessen der Programmaufbau leicht nachvollzogen werden kann.

5.1. Beispiel-Programm aus Kapitel 1

Wir beginnen wieder mit der einfachsten Form der Grafik, den Commodore-Grafik-Zeichen. Das erste Programm besteht aus den schon bekannten Beispielen des 1. Kapitels.

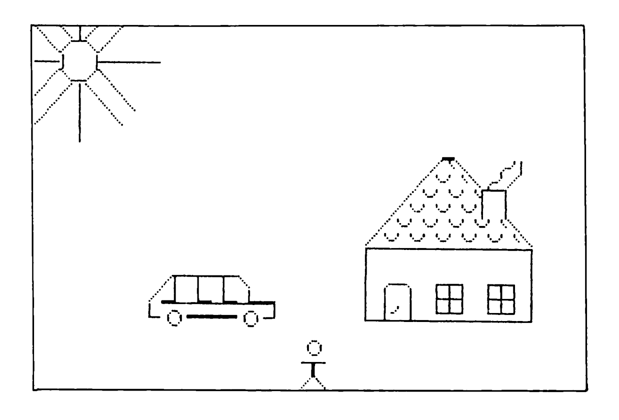


Abb. 5.1

```
10 REM.................................BEISPIEL KAPITEL 1
15 REM........................SONNE; HAUS; AUTO; STRICHMAENNCHEN;
18 REM
20 PRINT"[CLR]";:REM.............JETZT FOLGEN DIE 'CODIERTEN' ZEICHEN
30 REM.................................SONNE
40 REM
280 PRINTCHR$(109);CHR$(32);CHR$(109);CHR$(125);CHR$(110);
300 PRINTCHR$(32);CHR$(110)
320 PRINTCHR$(32);CHR$(109);CHR$(110);CHR$(163);CHR$(109);
340 PRINTCHR$(110):PRINTCHR$(96);CHR$(96);CHR$(116);CHR$(32);
360 PRINTCHR$(167);CHR$(96);CHR$(96);CHR$(96);CHR$(96);CHR$(96)
380 PRINTCHR$(32);CHR$(110);CHR$(109);CHR$(114);CHR$(110);
400 PRINTCHR$(109):PRINTCHR$(110);CHR$(32);CHR$(110);CHR$(125);
420 PRINTCHR$(109);CHR$(32);CHR$(109)
440 PRINTCHR$(32);CHR$(110);CHR$(32);CHR$(125);CHR$(32);
460 PRINTCHR$(109);CHR$(32);CHR$(109)
480 PRINTCHR$(110);CHR$(32);CHR$(32);CHR$(125);CHR$(32);CHR$(32);
500 PRINTCHR$(109)
520 PRINTCHR$(32);CHR$(32);CHR$(32);CHR$(125)
540 REM.................................HAUS
560 REM DIE VERWENDETEN TASTEN STEHEN IN KLAMMERN HINTER DER ZEILE
580 REM
600 PRINTTAB(32);"▁":REM                (P)
620 PRINTTAB(31);"╱╲   ╱| ":REM          (N,J,M,U,K,M)
640 PRINTTAB(30);"╱◡ ╲╱╱":REM            (N,J,K,M)
660 PRINTTAB(29);"╱◡ ◡ ┬┐":REM           (N,J,K,O,P)
680 PRINTTAB(28);"╱◡ ◡ ◡| |":REM         (N,J,K,H,N)
700 PRINTTAB(27);"╱◡ ◡ ◡ ◡ ╲":REM        (N,J,K,M)
720 PRINTTAB(26);"╱◡ ◡ ◡ ◡ ◡ ╲":REM      (N,J,K,M)
740 PRINTTAB(26);"┌────────────┐":REM   (O,Y,P)
760 PRINTTAB(26);"|            |":REM   (H,N)
780 PRINTTAB(26);"| ╭╮  ┬┬  ┬┬ |":REM   (H,U,*,I,A,R,S,N)
800 PRINTTAB(26);"| | | ┼┼  ┼┼ |":REM   (H,-,Q,+,W,N)
820 PRINTTAB(26);"| |╱| ┴┴  ┴┴ |":REM   (H,-,K,Z,E,X,N)
840 PRINTTAB(26);"─────────────":REM    (Y)
```

```
850 REM.................................FARBE AUTO
860 FORJ=0TO3:Q=55984+J*40
870 FORI=0TO10:POKEQ+I,3
880 NEXTI
890 NEXTJ
895 REM.................................BESTANDTEILE AUTO
900 POKE 1714,78:POKE 1715,79:POKE 1716,80
920 POKE 1717,119:POKE 1718,80:POKE 1719,119
940 POKE 1720,77:POKE 1753,78:POKE 1754,111
960 POKE 1755,76:POKE 1756,122:POKE 1757,111
980 POKE 1758,122:POKE 1759,111:POKE 1760,122
1000 POKE 1761,111:POKE 1762,111
1020 POKE 1793,76:POKE 1794,85:POKE 1795,73
1040 POKE 1796,111:POKE 1797,111:POKE 1798,111
1060 POKE 1799,111:POKE 1800,85:POKE 1801,73
1080 POKE 1802,122:POKE 1834,74:POKE 1835,75
1120 POKE 1840,74:POKE 1841,75
1160 REM.................................FARBE HAUS
1170 FOR J=0TO5: Q=55922+J*40
1180 FOR I=0TO12:POKE Q+I,8
1190 NEXT I
1200 NEXT J
1210 REM.................................FENSTER
1220 FOR J=0TO2: Q=56003+J*40
1230 FOR I=0TO10:POKE Q+I,5
1240 NEXT I
1250 NEXT J
1255 REM.................................DACH
1260 FOR J=0TO6: Q=55642+J*40
1270 FOR I=0TO12:POKE Q+I,2
1280 NEXT I
1290 NEXT J
1295 REM.................................SCHORNSTEIN
1300 POKE 55771,8:POKE 55772,8:POKE 55811,8:POKE 55812,8
1310 REM.................................RAUCH
1320 FOR J=0TO1: Q=55691+J*40
1330 FOR I=0TO3:POKE Q+I,1
1340 NEXT I
1350 NEXT J
```

```
1355 REM..................................FARBE SONNE
1360 FOR J=0TO7: Q=55296+J*40
1370 FOR I=0TO9:POKE Q+I,7
1380 NEXT I
1390 NEXT J
1400 FOR I=56136TO56295:POKE I,10:NEXT I:REM FARBE FIGUR
1420 R=1864
1440 :
1450 FOR J=0TO38
1460 FOR I=0TO3: Q=R+I*40:REM............FIGUR SETZEN
1480 READ A,B:POKE Q+J,A:POKE Q+J+1,B:NEXT I: RESTORE
1500 FOR Z=1TO250:NEXT Z:REM.............WARTESCHLEIFE
1520 FOR I=0TO3: Q=R+I*40:REM............FIGUR LOESCHEN
1540 POKE Q+J,96:POKE Q+J+1,96
1550 NEXT I
1560 NEXT J
1570 :
1580 GOTO 1440
2000 DATA 85,73,74,75,80,79,78,77

READY.
```

5.2. Balken-Diagramm aus 'eingebauten' Grafik-Zeichen

Eine praktische Anwendungsmöglichkeit dieser Grafik-Form ist die Darstellung beliebiger Zahlenwerte als Balken-Diagramme. Da hierfür keine hohe Auflösung der Grafik erforderlich ist, bietet es sich an, die "fest eingebauten" Grafik-Zeichen des Commodore 64 zu verwenden. Beim Gebrauch dieser Zeichen ist darauf zu achten, daß jeweils ein Zeichen ein Bildschirmkästchen einnimmt. Aus diesem Grund passen nicht alle Zeichen nahtlos aneinander.

In dem folgenden Programm wird das Zeichen mit der Code-Nr. 103 als linke, das Zeichen mit der Code-Nr. 101 als rechte und das Zeichen mit der Code-Nr 100 als obere Begrenzung benutzt (s. Kapitel 7, Commodore-Zeichen).

Der Farbspeicher wird übrigens diesmal nicht benötigt, um die einzelnen Balken sichtbar zu machen. In Zeile 420 wird durch POKE 53280,0 die Hintergrund- (und Rahmenfarbe) schwarz, die Vordergrundfarbe durch POKE 53281,1 weiß. Die Zeichen erscheinen auf dem Vordergrund dann in der Hintergrundfarbe.

Wenn Sie das Programm gestartet haben, sehen Sie die Ergebnisse der letzten Bundestagswahl in Form eines Balken-Diagramms auf dem Bildschirm. Durch eine Änderung der Werte in den DATA-Zeilen können Sie sich leicht Ihr persönliches Traumergebnis darstellen lassen.

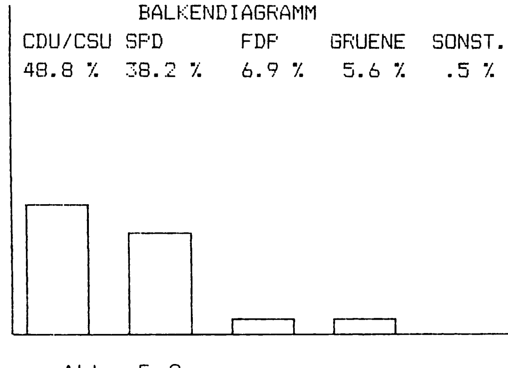
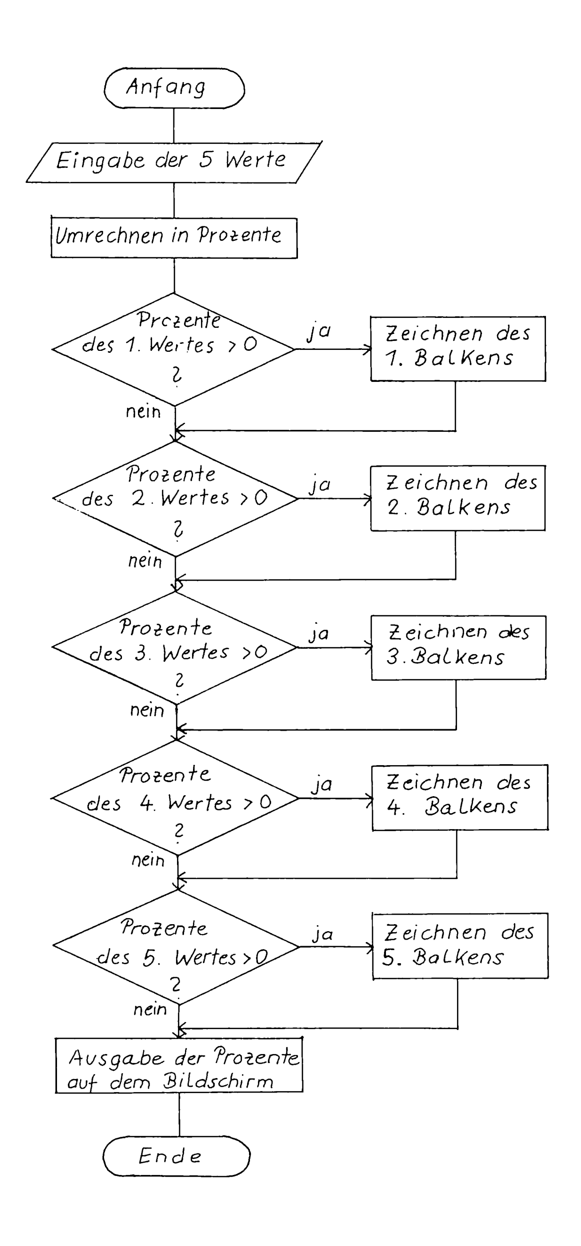


Abb. 5.2



Fluß - Diagramm : Balkendiagramme

```
20 REM................................BALKENDIAGRAMM
30 REM
140 PRINT
150 READ E1,E2,E3,E4,E5
380 REM..............DARSTELLUNG MIT COMMODORE GRAFIK
400 REM...............................X-KOORDINATE
410 PRINT"♥"
420 POKE 53280,0: POKE 53281,1
430 FOR I=1TO39
440 POKE 1984+I,119
450 NEXT I
460 REM...............................Y-KOORDINATE
470 FOR I=1TO23
480 POKE 1024+40*I,103
490 NEXT I
500 PRINTTAB(10);CHR$(18);" BALKENDIAGRAMM ";CHR$(146)
600 REM...............................BEGRENZUNGEN
605 REM...............................1.BALKEN
610 IF INT(E1/5)=0 THENGOTO785
620 FOR I=1TOINT(E1/5)
640 POKE 1946-40*(I-1),101
660 NEXT I
680 FOR I=1TO5
700 POKE 1946-40*(INT(E1/5))+I-1,100
720 NEXT I
740 FOR I=1TOINT(E1/5)
760 POKE 1950-40*(I-1),103
780 NEXT I
785 REM...............................2.BALKEN
790 IF INT(E2/5)=0 THENGOTO965
800 FOR I=1TOINT(E2/5)
820 POKE 1954-40*(I-1),101
840 NEXT I
860 FOR I=1TO5
880 POKE 1954-40*(INT(E2/5))+I-1,100
900 NEXT I
920 FOR I=1TOINT(E2/5)
940 POKE 1958-40*(I-1),103
960 NEXT I
```

```
965 REM................................3.BALKEN
970 IF INT(E3/5)=0 THENGOTO1135
980 FOR I=1TOINT(E3/5)
1000 POKE 1962-40*(I-1),101
1020 NEXT I
1040 FOR I=1TO5
1060 POKE 1962-40*(INT(E3/5))+I-1,100
1080 NEXT I
1100 FOR I=1TOINT(E3/5)
1120 POKE 1966-40*(I-1),103
1130 NEXT I
1135 REM................................4.BALKEN
1140 IF INT(E4/5)=0 THENGOTO1315
1160 FOR I=1TOINT(E4/5)
1180 POKE 1970-40*(I-1),101
1200 NEXT I
1220 FOR I=1TO5
1240 POKE 1970-40*(INT(E4/5))+I-1,100
1260 NEXT I
1280 FOR I=1TOINT(E4/5)
1300 POKE 1974-40*(I-1),103
1310 NEXT I
1315 REM................................5.BALKEN
1320 IF INT(E5/5)=0 THENGOTO1500
1340 FOR I=1TOINT(E5/5)
1360 POKE 1978-40*(I-1),101
1380 NEXT I
1400 FOR I=1TO5
1420 POKE 1978-40*(INT(E5/5))+I-1,100
1440 NEXT I
1470 FOR I=1TOINT(E5/5)
1480 POKE 1982-40*(I-1),103
1490 NEXT I
```

```
1500 PRINTTAB(2);"CDU/CSU SPD      FDP    GRUENE  SONST."
1520 PRINTTAB(1);E1;"%";
1540 PRINTSPC(1);E2;"%";
1560 PRINTSPC(2);E3;"%";
1580 PRINTSPC(2);E4;"%";
1590 PRINTSPC(2);E5;"%";
1600 GETA$: IFA$="" THENGOTO1600
1700 POKE 53280,254: POKE 53281,246: PRINT"": END
1800 DATA 48.8,38.2,6.9,5.6,.5

READY.
```

5.3. Farbiges Balken-Diagramm aus 'eingebauten' Grafik-Zeichen

In diesem Programm wird ein geschlossenes, farbiges Balken-Diagramm erzeugt. Dazu wird diesmal als Vordergrund- und Hintergrundfarbe (in Zeile 420) weiß gewählt. Die Balken entstehen durch Benutzung des inversen Cursors (Code-Nr. 224) zusammen mit einer Farbangabe für den jeweiligen Balken. Bei einer solchen Darstellungsart ist es, durch Wahl der Speicherplätze in Bildschirm- und Farbspeicher, möglich, die Abstände der Balken zueinander nach eigenen Vorstellungen zu verändern, sie sogar sich überschneiden zu lassen.

Ein weiterer Unterschied zum ersten Balken-Diagramm-Programm besteht darin, daß die darzustellenden Werte nicht in DATA-Zeilen stehen, sondern über INPUT (Zeile 160 bis 230) eingegeben werden.

Beide Programme sollen Ihnen nur als Anregung dienen. Sie sind natürlich nicht auf fünf Balken bzw. fünf Werte beschränkt; genauso können Sie statt der Prozente auch Mittelwerte oder andere statistische Daten berechnen lassen.

```
10 REM
20 REM................................BALKENDIAGRAMM
30 REM
100 PRINT"❐"
120 PRINT"***DARSTELLUNG VON FUENF EINZELWERTEN IN DIAGRAMMFORM**"
140 PRINT
145 REM................................EINGABE DER WERTE
160 INPUT"WERT 1";E1
180 INPUT"WERT 2";E2
200 INPUT"WERT 3";E3
220 INPUT"WERT 4";E4
230 INPUT"WERT 5";E5
240 REM................................BERECHNUNG DER SUMME
260 S=E1+E2+E3+E4+E5
270 IF S=0 THEN PRINT:PRINT"SIE HABEN NUR NULLEN EINGEGEBEN."
275 IF S=0 THEN PRINT"WIEDERHOLEN SIE MIT NEUEN WERTEN.":GOTO1750
280 REM................................BERECHNUNG DER PROZENTE
300 P1=E1*100/S
320 P2=E2*100/S
340 P3=E3*100/S
360 P4=E4*100/S
370 P5=E5*100/S
380 REM................................DARSTELLUNG MIT LEERZEICHEN
410 PRINT"❐"
415 REM................................BILDSCHIRM UMSCHALTEN
420 POKE53280,1:POKE53281,1
425 REM................................X-KOORDINATE
430 FOR I=1TO 39
440 POKE 1984+I,119
450 NEXT I
460 REM................................Y-KOORDINATE
470 FOR I=1TO23
480 POKE 1024+40*I,103
490 NEXT I
500 PRINTTAB(10);CHR$(18);" BALKENDIAGRAMM ";CHR$(146)
600 REM................................BEGRENZUNGEN
```

```
605 REM.................................1.BALKEN
610 IF INT(P1/5)=0 THEN GOTO740
620 FOR J=1TO INT(P1/5)
640 FOR I=1TO5
660 POKE 1946-40*(J-1)+I-1,224
680 POKE 56218-40*(J-1)+I-1,10
700 NEXT I
710 NEXT J
715 REM.................................2. BALKEN
740 IF INT(P2/5)=0 THEN GOTO860
760 FOR J=1TO INT(P2/5)
780 FOR I=1TO5
790 POKE 1954-40*(J-1)+I-1,224
800 POKE 56226-40*(J-1)+I-1,7
820 NEXT I
840 NEXT J
845 REM.................................3. BALKEN
860 IF INT(P3/5)=0 THEN GOTO1000
880 FOR J=1TO INT(P3/5)
900 FOR I=1TO5
920 POKE 1962-40*(J-1)+I-1,224
940 POKE 56234-40*(J-1)+I-1,5
960 NEXT I
980 NEXT J
985 REM.................................4.BALKEN
1000 IF INT(P4/5)=0 THEN GOTO1150
1020 FOR J=1TO INT(P4/5)
1040 FOR I=1TO5
1060 POKE 1970-40*(J-1)+I-1,224
1080 POKE 56242-40*(J-1)+I-1,5
1100 NEXT I
1120 NEXT J
1145 REM...............................5.BALKEN
1150 IFINT(P5/5)=0THENGOTO1325
1160 FOR J=1TOINT(P5/5)
1170 FOR I=1TO5
1180 POKE 1978-40*(J-1)+I-1,224
1200 POKE 56250-40*(J-1)+I-1,9
1220 NEXT I
1230 NEXT J
```

```
1325 REM..............................UEBERSCHRIFT
1500 PRINTTAB(2);"WERT 1  WERT 2  WERT 3  WERT 4  WERT 5"
1520 PRINTTAB(2);INT(P1);"%";
1540 PRINTSPC(4);INT(P2);"%";
1560 PRINTSPC(3);INT(P3);"%";
1580 PRINTSPC(3);INT(P4);"%";
1590 PRINTSPC(3);INT(P5);"%";
1600 GETA$:IFA$=""THENGOTO1600
1650 REM..........................UMSCHALTEN ZUM NORMALEN MODUS
1700 POKE 53280,254:POKE 53281,246:PRINT"□"
1750 END

READY.
```

5.4. Das Sprites-Aquarium

Wie Sie (hoffentlich!) aus dem folgenden Programm ersehen können, sollen alle 8 möglichen Sprites - in Gestalt von Fischen - auf dem Bildschirm erscheinen (Zeile 40).

In Zeile 60 werden die Sprites-Daten aus den DATA-Zeilen gelesen und in die entsprechenden Speicherplätze geschrieben.

In der nächsten Zeile (65) werden die Sprites mit den Nummern 0,1,6 und 7 in X- und Y-Richtung vergrößert.

Dann ist es soweit: die Sprites-Fische ziehen über den Bildschirm (Zeile 70 - 115):

In Zeile 70 wird durch die Schrittweite der FOR-Schleife die Geschwindigkeit der Sprites festgelegt.

In den Zeilen 75 bis 85 werden die X-Koordinaten der Sprites in die jeweiligen Speicher geschrieben, in den Zeilen 90 bis 100 geschieht das gleiche mit den Y-Koordinaten.

Vielleicht experimentieren Sie noch ein wenig mit diesem Programm! Sie können z.B. einfach die Geschwindigkeit ändern, indem Sie in Zeile 70 eine andere Schrittweite hinter dem Schlüsselwort STEP angeben. Durch andere Koordinaten können Sie die Lage der Fische verändern, Sie können Fische vergrößern, schmal oder breit werden lassen, ja, und dann natürlich können Sie auch - am besten mit Hilfe des folgenden Programms - ganz neue Fische oder sonstige Bewohner für Ihr Aquarium erfinden!

```
10 REM..................................AQUARIUM
15 PRINT"♥"
30 V=53248:REM...........................VIDEOCHIP
40 POKE V+21,255:REM.....................ALLE SPRITES
45 POKE 2040,14:POKE 2041,14
50 POKE 2042,14:POKE 2043,14:POKE 2044,14
55 POKE 2045,14:POKE 2046,13:POKE 2047,13
60 FOR N=0TO125:READ Q:POKE832+N,Q:NEXT N
65 POKE V+23,195:POKE V+29,195
70 FOR X=1TO200STEP1:REM.................BEWEGUNG
75 POKE V+0,X:POKE V+2,X
80 POKE V+4,X:POKE V+6,X:POKE V+8,X
85 POKE V+10,X:POKE V+12,200-X:POKE V+14,200-X
90 POKE V+1,100:POKE V+3,180-0.8*X
95 POKE V+5,230-ABS(10*SIN(X)):POKE V+7,200-ABS(RND(X))*10
100 POKEV+9,100+7.5*SQR(X):POKEV+11,180:POKEV+13,200-X:POKEV+15,X
110 NEXTX
115 GOTO70
120 DATA 6,0,0,6,0,0,0,6,0,56,14,0,48,31,0
125 DATA 0,127,0,97,255,128,103,63,193
130 DATA 15,63,227,31,255,247,207,255,255,199
135 DATA 255,255,15,255,255,31,255,247,15
140 DATA 255,227,7,255,193,1,255,128,0,127,0,0,30,0,0,14,0,0,4,0
145 REM.................................2.FISCH
150 DATA 224,0,0,0,0,0,0,0,112,0,0,60,0,0,30
155 DATA 0,1,255,0,3,146,192,6,109,176,140
160 DATA 146,92,217,36,132,242,73,246,228
165 DATA 146,252,242,73,246,217,36,134,140
180 DATA 146,92,6,109,176,3,146,192,1,255,0
185 DATA 0,255,0,0,120,0,0,224,0

READY.
```

5.5. Sprites-Erstellen auf dem Bildschirm

Vielleicht haben Sie sich bei der doch sehr mühseligen Berechnung der Sprite-Figuren auch schon gefragt, wozu es eigentlich Computer gibt? Eine derartig schematisierte Rechnerei bietet sich natürlich geradezu an, von einem Computer ausgeführt zu werden.

Im folgenden wird nun ein Programm beschrieben, daß es dem Computer überläßt, die notwendigen Sprite-Daten zu berechnen. Die gewünschte Sprite-Figur wird mit den üblichen Cursor-Bewegungen direkt auf dem Bildschirm konstruiert, indem man innerhalb eines vorgegebenen Rahmens ein - stark vergrößertes - Bild der Sprite-Figur zeichnet.

Die Handhabung des Programmes ist sehr einfach. Man setzt die erforderlichen Punkte in einen markierten Bildschirmbereich, der aus 21 Zeilen mit jeweils 24 Spalten besteht (Sprite-Größe). (Wir haben dafür den geschlossenen Kreis auf der 'Q'-Taste benutzt, Sie können aber auch jedes beliebige andere Zeichen verwenden.) Ist man mit seinem Werk zufrieden, so kann man es anschließend - nach Drücken von RETURN und einer kurzen Pause zum Rechnen - in Originalgröße als "echtes" Sprite auf dem Bildschirm sehen.

Das so erstellte Sprite läßt sich außerdem noch vergrößern oder auch wieder in seiner ursprünglichen Größe darstellen. Dazu stehen Ihnen folgende Möglichkeiten - jeweils durch Drücken einer bestimmten Taste - zur Auswahl:

- B Breit. Vergrößert das Sprite in X-Richtung.
- H Hoch. Vergrößert in Y-Richtung.
- G Groß. Vergrößert in X- und Y-Richtung.
- O Original. Darstellung in Originalgröße.
- M Mehrfarbig. Falls Sie ein mehrfarbiges Sprite entworfen haben.
- V Verändern. Verzweigung zum Programmanfang, um den Sprite-Entwurf weiter zu verändern.
- D Daten. Ausgabe der Daten und Programmende.

Ist man mit dem Ergebnis noch nicht ganz zufrieden, kann man also nach der Eingabe von 'V' das Sprite weiter verändern.

```
10 REM.................................SPRITE-HILFSPROGRAMM
15 PRINT"♥":REM.1.BILDSCHIRM FUR PROGRAMMERKLAERUNG
20 PRINTTAB(10);CHR$(18);" SPRITE-ENTWERFEN "
25 PRINT:PRINT" MIT DIESEM PROGRAMM KOENNEN SIE EIN    ";
30 PRINT" SPRITE DEFINIEREN UND SICH DIE WERTE    ";
35 PRINT"AUSRECHNEN LASSEN."
40 PRINT:PRINT" ES ERSCHEINT EIN RAHMEN AUF DEM BILD-  ";
45 PRINT" SCHIRM, IN DEM SIE DAS SPRITE ENT-"
50 PRINT" WERFEN KOENNEN."
65 PRINT"  NACH 'RETURN' ERSCHEINT DAS SPRITE."
70 PRINT:PRINT"DANACH HABEN SIE FOLGENDE MOEGLICHKEITEN"
80 PRINT"  'B': VERBREITERT DAS SPRITE"
90 PRINT"  'H': LAESST ES HOEHER WERDEN"
100 PRINT"  'G': VERGROESSERT IN BEIDE RICHTUNGEN"
110 PRINT"  'O': STELLT ORIGINALGROESSE HER"
115 PRINT"  'M': FUER EIN MEHRFARBIGES SPRITE"
120 PRINT"  'V': ENTWURF KANN GEAENDERT WERDEN"
130 PRINT"  'D': DATENAUSGABE UND PROGRAMMENDE"
140 PRINT:PRINT" DRUECKEN SIE RETURN UND DER RAHMEN "
150 PRINT" ERSCHEINT. VIEL SPASS!"
160 REM.................................2.BILDSCHIRM
180 INPUTA$
190 PRINT"♥"
195 REM.................................TEXT ZUM RAHMEN
200 PRINTTAB(27);"1.SPRITE ENT-";
201 PRINT SPC(29);"WERFEN"
205 PRINTTAB(27);"2.'RETURN'"
210 PRINTTAB(27);"  ZUM SEHEN"
215 PRINTTAB(27);"3.AUSWAHL:"
220 PRINTTAB(27);" 'B' BREIT"
225 PRINTTAB(27);" 'H' HOCH"
230 PRINTTAB(27);" 'G' GROSS"
235 PRINTTAB(27);" 'O' ORIGINAL";
237 PRINTSPC(27);" 'M' MEHRF."
240 PRINTTAB(27);" 'V' AENDERN"
245 PRINTTAB(27);"     (WIEDER"
250 PRINTTAB(27);"      ZU 1.)"
255 PRINTTAB(27);" 'D' DATEN";
256 PRINT SPC(34);"UND ENDE"
```

```
260 REM.................................RAHMEN
270 FORI=1026TO1049:POKEI,100:NEXTI
275 FORI=55298TO55321:POKEI,14:NEXTI
280 FORI=1065TO1865STEP40:POKEI,103:NEXTI
285 FORI=55337TO56137STEP40:POKEI,14:NEXTI
290 FORI=1090TO1890STEP40:POKEI,101:NEXTI
295 FORI=55362TO56162STEP40:POKEI,14:NEXTI
300 FORI=1906TO1929:POKEI,99:NEXTI
305 FORI=56178TO56211:POKEI,14:NEXTI
320 INPUTA$
340 REM.................................BERECHNUNGEN
350 Q=0:Y=0:FORX=0TO20
400 FORI=0TO7:IFPEEK(1066+40*X+I)=32THENA=0:GOTO500
450 A=1
500 Q=Q+A*(2↑(7-I))
550 NEXTI
600 POKE960+Y,Q
650 Q=0:Y=Y+1:FORI=0TO7
700 IFPEEK(1074+40*X+I)=32THENA=0:GOTO800
750 A=1
800 Q=Q+A*(2↑(7-I))
850 NEXTI
900 POKE960+Y,Q
950 Q=0:Y=Y+1:FORI=0TO7
1000 IFPEEK(1082+40*X+I)=32THENA=0:GOTO1100
1050 A=1
1100 Q=Q+A*(2↑(7-I))
1150 NEXTI
1200 POKE960+Y,Q
1250 Q=0:Y=Y+1
1300 NEXTX
1500 V=53248
1550 POKEV+21,4:POKE2042,15
1570 POKEV+23,0:POKEV+29,0
1600 POKEV+4,250:POKEV+5,180
1650 POKEV+37,7:POKEV+38,5:POKEV+41,14
1700 GETA$:IFA$=""THENGOTO1700
```

```
1730 REM................................TASTATURABFRAGE
1750 IFA$="O"THENPOKEV+21,4:POKE2042,15:POKEV+23,0:POKEV+29,0
1760 IFA$="O"THENPOKEV+28,0:GOTO1700
1770 IFA$="M"THENPOKEV+28,4:GOTO1700
1800 IFA$="B"THENPOKEV+29,4:POKEV+23,0:GOTO1700
1850 IFA$="H"THENPOKEV+29,0:POKEV+23,4:GOTO1700
1900 IFA$="G"THENPOKEV+29,4:POKEV+23,4:GOTO1700
1950 IFA$="V"THENPOKEV+21,0:GOTO300
2000 IFA$="D"THENGOTO2250
2040 REM................................RUECKSPRUNG
2050 GOTO1700
2250 PRINT"♥"
2300 FORI=960TO1022:PRINTPEEK(I),:NEXTI
2500 END

READY.
```

5.6. Sprites-Eisenbahn in hochauflösender Grafik

Das nächste Programm nutzt sowohl die Möglichkeiten der Sprites als auch die der hochauflösenden Grafik.

Auf einem "Gleisnetz", dargestellt in der hochauflösenden Grafik, bewegt sich ein Triebwagen (das mehrfarbige Sprite aus dem Beispiel 2.3). Mit Hilfe der Cursor-Tasten läßt sich der Wagen steuern:

→ Der Wagen fährt auf dem äußeren Gleis rechts-

← bzw. linksherum.

↓ Jetzt wird das innere Gleis (rechtsherum) benutzt,

↑ und nun fährt der Triebwagen auf dem inneren Gleis links herum.

Die Abfrage, welche Taste gedrückt wurde, steht in Zeile 860 - 940. Wird keine dieser Tasten gedrückt, bleibt die Lok stehen.

Auch dieses Programm können Sie natürlich Ihren Vorstellungen entsprechend erweitern!

```
100 REM................................EISENBAHN
200 PRINT"□"
300 REM...........................SPRITE ZEIGEN UND ERKLAERUNGEN
400 V=53248
410 POKEV+21,4:POKEV+28,4:POKE2042,13
420 POKEV+37,7:POKEV+38,0:POKE2042,13
430 FORI=0TO62:READQ:POKE832+I,Q:NEXTI
440 POKEV+41,2:POKEV+4,30:POKEV+5,50
450 PRINT:PRINTCHR$(18);TAB(6)" LIEBE EISENBAHN-FREUNDE, "
460 PRINT:PRINTCHR$(146);"  ICH BIN EIN TRIEBWAGEN UND";
470 PRINT" LASSE        MICH BEWEGEN. DAZU BRAUCHEN SIE NUR
480 PRINT"DIE CURSOR-TASTEN."
490 PRINT:PRINT"  CURSOR RECHTS: ICH FAHRE AUF DEM"
500 PRINT"               AEUSSEREN GLEIS"
510 PRINT"               RECHTSHERUM."
520 PRINT"  CURSOR LINKS:  NUN FAHRE ICH DORT  "
530 PRINT"               LINKSHERUM."
540 PRINT"  CURSOR UNTEN:  JETZT BENUTZE ICH DAS"
550 PRINT"               INNERE GLEIS,"
560 PRINT"  CURSOR OBEN:   UND AUCH HIER KANN ICH"
570 PRINT"               LINKSHERUM FAHREN."
580 PRINT:PRINT"  DRUECKEN SIE KEINE ODER IRGENDEINE"
590 PRINT"  ANDERE TASTE, SO BLEIBE ICH STEHEN."
600 PRINT:PRINT:PRINT"  GEBEN SIE JETZT 'RETURN' EIN, BE-"
610 PRINT"  GINNT DAS PROGRAMM MIT DEM AUFBAU"
620 PRINT"  DES GLEISNETZES. VIEL SPASS!"
630 INPUTA$
640 POKE V+21,0
650 POKE V+17,59:POKE V+24,24
660 FOR I=1024TO2023:POKE I,14:NEXT I
670 FOR I=8192TO16191:POKE I,0:NEXT I
```

```
675 REM.................................GLEISE
680 FOR I =-πTOπSTEPπ/180
690 X=130+120*COS(I)
700 Y=100+90*SIN(I)
710 GOSUB 60000
720 NEXT I
740 FOR I=-πTOπ STEPπ/180
750 X=130+120*COS(I)
760 Y=100+50*SIN(I)
770 GOSUB 60000
780 NEXTI
785 REM.................................BAHNHOF
790 FOR I=0TO4
800 FOR Z=0TO23:READQ:POKE 11656+I*320+Z,Q:NEXT Z
810 NEXT I
820 POKE V+41,2:POKE V+28,4:REM..........WAGEN ERSCHEINT
830 POKE V+4,255:POKE V+5,135:POKE V+21,4
840 FOR I=0TO2*πSTEPπ/180
850 GET Z$
860 IF Z$=""THEN GOTO 850
870 Z%=ASC(Z$)
880 IF Z%=29 THEN X=135+120*COS(I):Y=135+90*SIN(I)
890 IF Z%=17 THEN X=135+120*COS(I):Y=135+50*SIN(I)
900 IF Z%=157 THEN I=I-π/90:X=135+120*COS(I)
910 IF Z%=157 THEN Y=135+90*SIN(I)
920 IF Z%=145 THEN I=I-π/90:X=135+120*COS(I)
930 IF Z%=145 THEN Y=135+50*SIN(I)
940 IF NOT(Z%=29ORZ%=17ORZ%=157ORZ%=145)THEN GOTO 850
950 POKE V+4,X
960 POKE V+5,Y
970 NEXT I
980 GOTO 840
990 RESTORE
```

```
4000 REM...............................DATEN TRIEBWAGEN
4010 DATA 0,0,0,0,0,0,0,0,0,21,85,84,85,85,84,80,20,5,80
4020 DATA 20,5,80,20,5,80,20,5,80,20,5,170,170,170,170
4030 DATA 170,170,170,170,170,130,170,130,140,170,50,63
4040 DATA 40,252,63,0,252,63,0,252,12,0,48,0,0,0,0,0,0
5000 REM...............................DATEN BAHNHOF
5010 DATA 255,128,128,128,128,128,128,128,255,0,0,0,0,0,0,0
5020 DATA 255,1,1,1,1,1,1,1
5030 DATA 128,128,128,128,128,128,128,128,0,36,36,36,60,36,36,36
5040 DATA 1,1,1,1,1,1,1,1,128,128,128,128,128,128,128,128
5050 DATA 0,0,32,32,56,36,36,56,1,1,1,1,1,1,1,1
5060 DATA 128,128,128,128,128,128,128,128,0,0,24,32,32,56,32,32
5070 DATA 1,1,1,1,1,1,1,1,128,128,128,128,128,128,128,255
5080 DATA 32,0,0,0,0,0,0,255,1,1,1,1,1,1,1,255
60000 REM...............................ZEICHNEN
60020 XK=8*INT(X/8)
60040 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60060 EX=2↑(7-INT((X/8-INT(X/8))*8))
60080 S=8192+XK+YK
60100 POKE S,PEEK(S) OR EX
60120 RETURN

READY.
```

5.7. BASIC-Uhr in hochauflösender Grafik

Sie erinnern sich an die Definition der Kreisfunktion? (Kap. 3.2.) Hier ist sie noch einmal in der Parameterform angegeben, also nach den beiden Koordinaten X und Y aufgelöst:

X = XM + R * COS(PHI),   Y = YM + R * SIN(PHI)

Nachdem wir durch die ersten Anweisungen die Grafik vorbereitet haben, wollen wir jetzt mit Hilfe dieser Funktion ein Ziffernblatt für unsere Uhr zeichnen. (Vgl. zu den folgenden Beschreibungen das Flußdiagramm: BASIC-Uhr.)

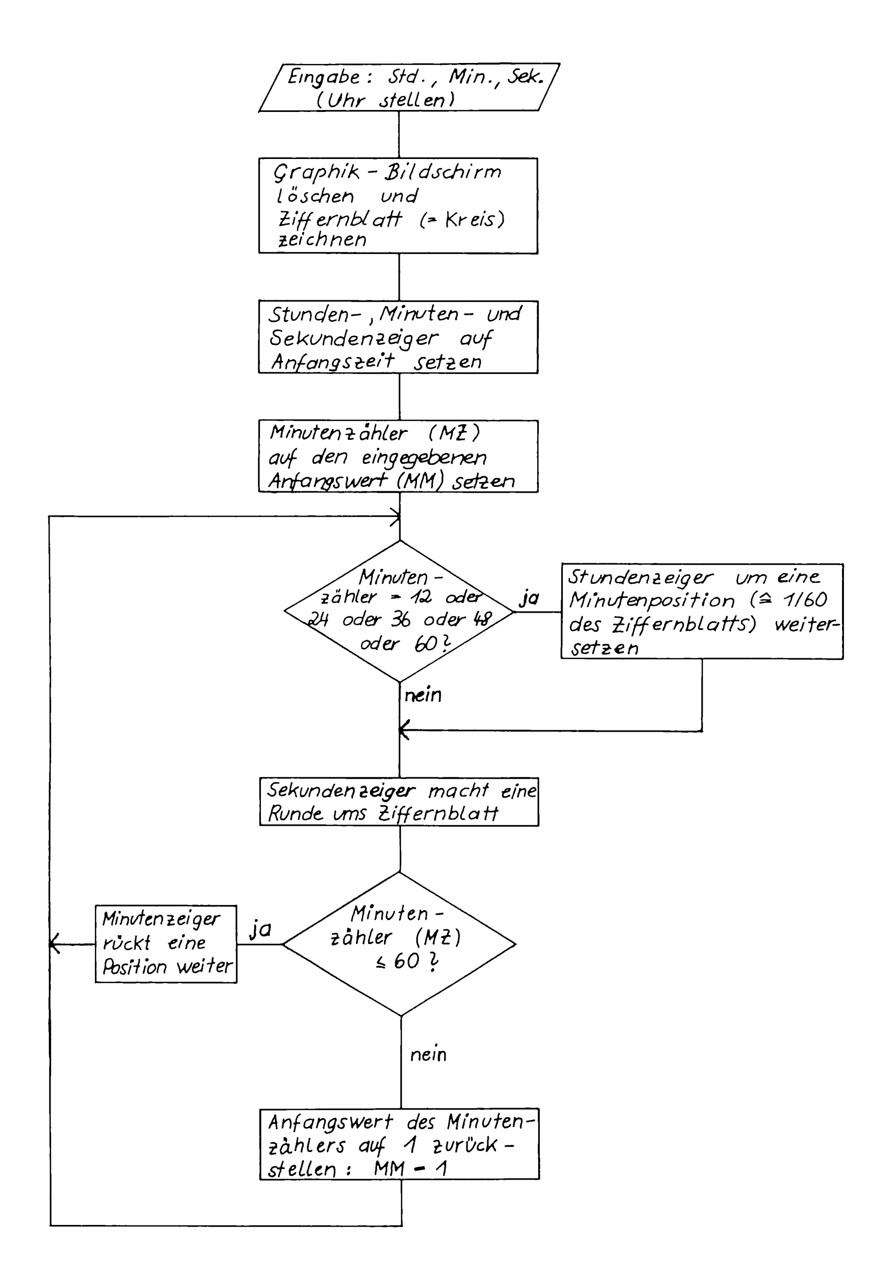
Damit sich die Uhr in der Mitte des Bildschirms befindet, wählen wir als Koordinaten XM = 160 und YM = 100. Wenn wir den Kreis durch nur 60 Punkte darstellen, erhalten wir sofort eine Einteilung des Ziffernblattes in 60 Minuten-Einheiten. Das Zeichnen dieser 60 auf einer Kreisbahn liegenden Punkte geschieht durch die Programmzeilen 200 - 540 (als Radius wurde der Wert 95 gewählt, um den Bildschirm fast ganz auszunutzen).

Anschließend wird die Uhr gestellt, d.h. Stunden-, Minuten- und Sekundenzeiger werden auf ihre Anfangspositionen gesetzt (555 - 1010). Zum Zeichnen des Stunden- und Minutenzeigers werden jeweils die Unterprogramme zum Zeichnen einer Geraden bei Zeile 5000 (bzw. 8000, wenn X1 = X2 ist) aufgerufen.

Soll ein Zeiger auf dem Zifferblatt vorrücken, muß der alte Zeiger gelöscht und ein neuer gezeichnet werden. Die entsprechenden Unterprogramme zum Löschen der Zeiger beginnen bei Zeile 7000 bzw. 9000.

Im Gegensatz zum Stunden- und Minutenzeiger besteht der Sekundenzeiger (ebenso wie der Mittelpunkt der Uhr) aus einem Sprite. Die Daten dieser Sprites werden in Zeile 542 - 546 eingelesen.

Der weitere Ablauf des Programms geht aus dem Flußdiagramm hervor. (Noch ein Hinweis: in Zeile 1730 beginnt der Sekundenzeiger nach jeder Minute mit dem Anfangswert 5. Dies geschieht deshalb, damit er die durch das Zeichnen des Minutenzeigers vergangene Zeit wieder 'aufholt'.)



Fluß-Diagramm: Basic-Uhr

```
20 REM...................................BASIC-UHR
30 REM
100 V=53248:REM..............ANFANGSADRESSE DES VIDEOCHIP
120 INPUT"STUNDE, MINUTE, SEKUNDE";HH,MM,SS
140 POKE V+17,59: POKE V+24,24
160 FOR I=1024TO2023: POKE I,1: NEXTI
180 FOR I=8192TO16383: POKEI,0: NEXTI: REM SPEICHER LOESCHEN
190 REM
200 REM.................................ZIFFERNBLATT
210 REM
220 FOR N=1TO60
240 RAD=RAD+(2*π)/60
260 X=COS(RAD)*95+160
280 Y=SIN(RAD)*95+100
300 GOSUB 6000
320 X=COS(RAD)*94+160
340 Y=SIN(RAD)*94+100
360 GOSUB 6000
380 IF (N/5-INT(N/5))<>0 THEN GOTO 540
390 REM.................................5 MIN. EINTEILUNG
400 X=COS(RAD)*93+160:Y=SIN(RAD)*93+100:GOSUB 6000
420 X=COS(RAD)*92+160:Y=SIN(RAD)*92+100:GOSUB 6000
440 X=COS(RAD)*91+160:Y=SIN(RAD)*91+100:GOSUB 6000
450 REM.................................VIERTELSTUNDENEINTEILUNG
460 IF(N/15-INT(N/15))<>0 THEN GOTO 540
480 X=COS(RAD)*90+160:Y=SIN(RAD)*90+100:GOSUB 6000
500 X=COS(RAD)*89+160:Y=SIN(RAD)*89+100:GOSUB 6000
520 X=COS(RAD)*88+160:Y=SIN(RAD)*88+100:GOSUB 6000
540 NEXT N
542 REM.........SPRITES FUER SEKUNDENZEIGER UND UHR-MITTELPUNKT
545 POKE V+21,8: POKE 2043,14
546 FOR Z=0TO125: READ Q: POKE 832+Z,Q:NEXTZ
550 POKE V+6,179: POKE V+7,140
555 REM...................STUNDENZEIGER AUF ANFANG SETZEN
580 SD=(2*π)/60*(HH*5-15+INT(MM/12))
600 XH=COS(SD)*50+160
620 XH=INT(XH/5)*5
640 YH=SIN(SD)*50+100
660 YH=INT(YH/5)*5
680 X1=160: Y1=100: X2=XH: Y2=YH
700 IF X2=X1 THEN GOSUB 8000: GOTO 740
720 GOSUB  5000
```

```
740 REM..................MINUTENZEIGER AUF ANFANG
760 MIN=(2*π)/60*(MM-15)
780 XM=COS(MIN)*75+160
800 XM=INT(XM/5)*5
820 YM=SIN(MIN)*75+100
840 YM=INT(YM/5)*5
850 X1=160:Y1=100:X2=XM:Y2=YM
860 IFX2=X1 THEN GOSUB 8000: GOTO 900
880 GOSUB 5000
900 REM........................SEKUNDENZEIGER AUF ANFANG
910 POKE 2042,13: POKE V+41,0
920 SEK=(2*π)/60*(SS-15)
940 XS=COS(SEK)*80+171
980 YS=SIN(SEK)*80+140
1010 POKE V+21,12: POKE V+4,XS: POKE V+5,YS
1020 :
1080 REM............................HAUPTPROGRAMM (MINUTENZEIGER)
1090 :
1100 FOR MZ=MMTO60
1120 IF (MZ=12 OR MZ=24 OR MZ=36 OR MZ=48 OR MZ=60) THEN GOSUB 1800
1140 GOSUB 1440
1160 X1=160: Y1=100: X2=XM: Y2=YM
1180 IF X1=X2 THEN GOSUB 9000
1190 IF X1=X2 THEN GOTO 1240
1200 GOSUB 7000
1220 IF(COS(MIN)=COS(SD)ANDSIN(MIN)=SIN(SD))THENX2=XH
1225 IF(COS(MIN)=COS(SD)ANDSIN(MIN)=SIN(SD))THENY2=YH
1230 IF(COS(MIN)=COS(SD)ANDSIN(MIN)=SIN(SD))THENGOSUB5000
1240 MIN=MIN+(2*π)/60
1260 XM=COS(MIN)*75+160
1280 XM=INT(XM/5)*5
1300 YM=SIN(MIN)*75+100
1320 YM=INT(YM/5)*5
1340 X1=160: Y1=100: X2=XM: Y2=YM
1360 IF X1=X2 THEN GOSUB 8000
1370 IF X1=X2 THEN GOTO 1400
1380 GOSUB 5000
1400 NEXT MZ
1420 MM=1: SS=SS+1
1430 GOTO 1100
```

```
1440 REM................................SEKUNDENZEIGER
1460 FOR SZ=SSTO59 STEP1
1480 X1=160: Y1=100: X2=XS: Y2=YS
1520 POKE V+21,8
1580 SEK=SEK+(2*π)/60
1600 XS=COS(SEK)*80+171
1640 YS=SIN(SEK)*80+140
1660 POKE V+21,12: POKE V+4,XS: POKE V+5,YS
1700 FOR PAUSE=1TO305: NEXT PAUSE
1720 NEXT SZ
1730 SS=5: SEK=(2*π)/60*(SS-15)
1740 RETURN
1800 REM.....................UNTERPROGRAMM STUNDENZEIGER
1820 X1=160: Y1=100: X2=XH: Y2=YH
1840 IF X1=X2 THEN GOSUB 9000
1850 IF X1=X2 THEN GOTO 1880
1860 GOSUB 7000
1880 SD=SD+(2*π)/60
2000 XH=COS(SD)*50+160
2010 XH=INT(XH)
2020 YH=SIN(SD)*50+100
2030 YH=INT(YH)
2040 X1=160:Y1=100:X2=XH:Y2=YH
2060 IF X2=X1 THEN GOSUB 8000
2080 IF X2=X1 THEN GOTO 3020
3000 GOSUB 5000
3020 RETURN
5000 REM...............................LINIE VON X1,Y1 NACH X2,Y2
5020 IF (X2-X1)<0 THEN X3=-(ABS(X2-X1))/20: GOTO 5080
5040 X3=(ABS(X2-X1))/20
5080 FOR X=X1TOX2 STEP X3
5100 Y=((X-X1)/(X2-X1))*(Y2-Y1)+Y1
5120 XK=8*INT(X/8)
5140 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
5160 EX=2↑(7-INT((X/8-INT(X/8))*8))
5180 S=8192+YK+XK
5200 POKE S,PEEK(S) OR EX
5210 NEXT X
5220 RETURN
```

```
6000 REM...........................................PUNKT X,Y
6020 XK=8*INT(X/8)
6040 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
6060 EX=2↑(7-INT((X/8-INT(X/8))*8))
6080 S=8192+YK+XK
6100 POKE S,PEEK(S) OR EX
6120 RETURN
7000 REM.......................................LINIE LOESCHEN
7020 IF (X2-X1)<0 THEN X3=-(ABS(X2-X1))/20: GOTO 7080
7040 X3=(ABS(X2-X1))/20
7080 FOR X=X2TOX1-1 STEP-X3
7090 Y=((X-X1)/(X2-X1))*(Y2-Y1)+Y1
7100 XK=8*INT(X/8)
7120 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
7140 EX=2↑(7-INT((X/8-INT(X/8))*8))
7160 S=8192+YK+XK
7180 POKE S,0
7200 NEXT X
7220 RETURN
8000 REM....................................LINIE MIT X=CONST.
8010 X3=ABS(Y1-Y2)/20
8020 IFY2<Y1THENX3=-X3
8040 FOR Y=Y1TOY2 STEP X3
8060 X=X1
8080 XK=8*INT(X/8)
8100 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
8120 EX=2↑(7-INT((X/8-INT(X/8))*8))
8140 S=8192+YK+XK
8160 POKE S,PEEK(S) OR EX
8170 NEXT Y
8180 RETURN
9000 REM..............................LINIE MIT X=CONST. LOESCHEN
9020 X3=ABS(Y1-Y2)/20
9030 IFY2<Y1THENX3=-X3
9060 FOR Y=Y2TOY1 STEP-X3
9080 X=X1
9100 XK=8*INT(X/8)
9120 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
9140 EX=2↑(7-INT((X/8-INT(X/8))*8))
9160 S=8192+YK+XK
9180 POKE S,0
9200 NEXTY
9220 RETURN
```

```
9999 REM..............................DATEN FUER SEKUNDENZEIGER
10000 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
10010 DATA 0,0,60,0,0,126,0,0,126,0,0,60,0,0,0,0,0,0,0,0,0,0
10020 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
10030 REM.............................DATEN FUER UHRMITTELPUNKT
10040 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,127,0,0
10060 DATA 127,0,0,127,0,0,127,0,0,127,0,0,127,0,0,127,0,0,0,0
10080 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

READY.
```

5.8. Darstellung mathematischer Funktionen in hochauflösender Grafik

Mit diesem Programm können Sie viele mathematische Funktionen auf dem Bildschirm darstellen. Voraussetzung ist allerdings, daß zu jedem X-Wert nur ein Y-Wert gehört (also nicht gleichzeitig ein positives und ein negatives Y). Für Funktionen, die diese Voraussetzung nicht erfüllen, können Sie aber immerhin den Absolutbetrag (ABS(Y)) zeigen lassen.

In den Zeilen 100 - 160 werden die X- und die Y-Achse gezeichnet, in den Zeilen 170 - 295 die Maßstabseinteilung. In Zeile 380 steht die Funktionsgleichung, die Sie - mit den erwähnten Einschränkungen - nach Belieben verändern können. Durch Änderung der Schrittweite (Zeile 310) können Sie sich die Funktion dichter oder mit weniger Punkten ausgeben lassen.

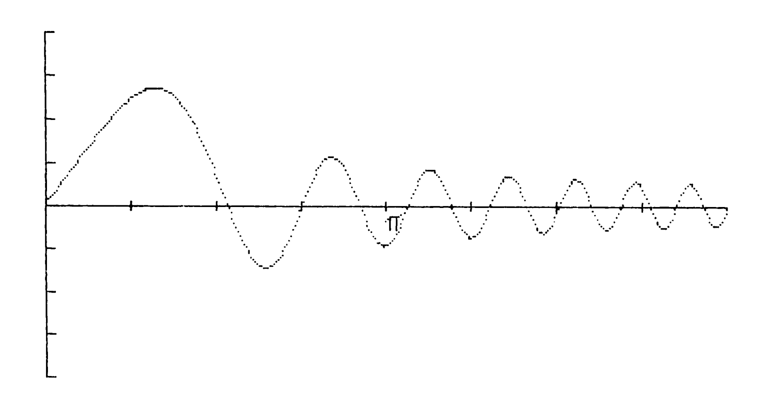
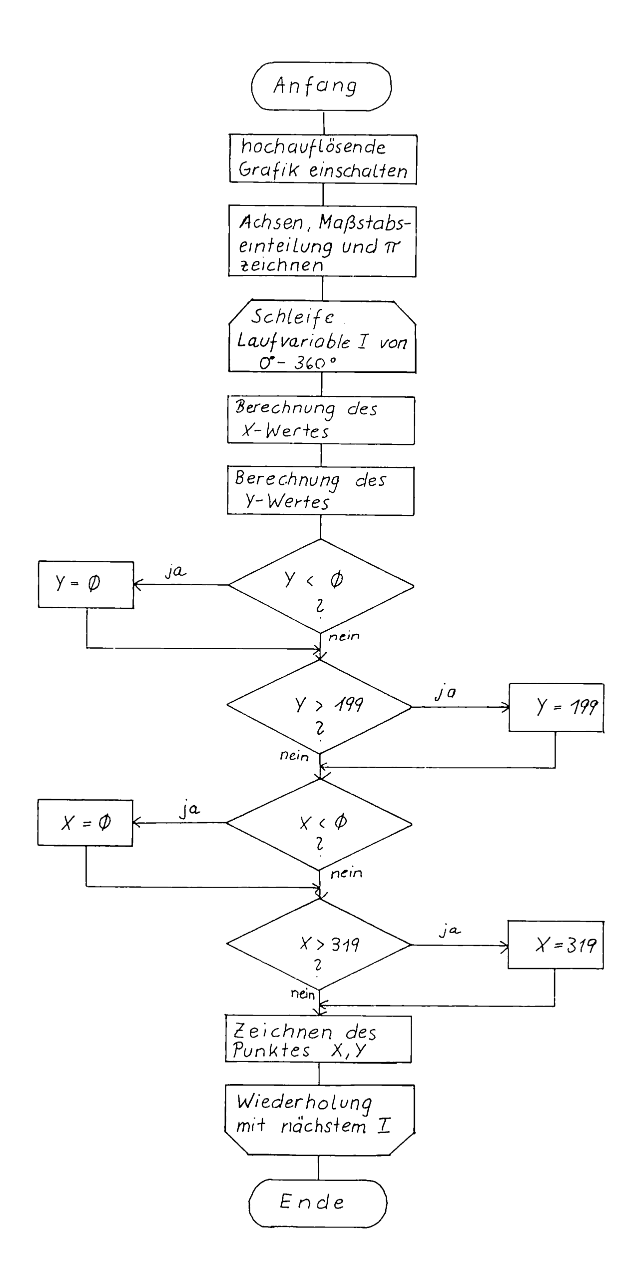


Abb. 5.6



Fluß - Diagramm : Funktionen

```
10 REM..................................FUNKTIONEN
20 REM..................................GRAPHIK EINSCHALTEN
30 V=53248
40 POKE V+17,59:POKE V+24,24
50 FOR I=1023TO2024:POKE I,14:NEXTI
60 FOR I=8192TO16383:POKE I,0:NEXTI
90 REM..................................ACHSEN U.EINTEILUNG
100 FOR X=0TO319
120 Y=100:GOSUB 60000
130 NEXT X
140 FOR Y=20TO180
150 X=0: GOSUB 60000
160 NEXT Y
170 FOR X=0TO319 STEP 40
180 FOR Y=98 TO 102
190 GOSUB 60000
200 NEXT Y
240 NEXT X
250 FOR Y=20TO180 STEP 20
260 FOR X=0TO4
270 GOSUB 60000
280 NEXT X
290 NEXT Y
295 GOSUB 1000:REM.......................π SETZEN
300 R1=1:R2=360:REM..................ANFANGS-U.ENDBEDINGUNG
310 FOR I=R1TOR2 STEP 3
320 IF ABS(R1)>=ABS(R2)THEN R=ABS(R1)
330 IF ABS(R2)>=ABS(R1)THEN R=ABS(R2)
340 S=70*4.57/R
350 X=I
360 X=X*3.14159/180
370 IF X=0THEN X=0.00001
375 REM.................................FUNKTION WAEHLBAR
380 Y1=SIN(X)*COS(4*X)
390 Y=-Y1*20+100
410 X=I*S
415 REM.................................WERTE ZULAESSIG ?
430 GOSUB 2000
435 REM.................................ZEICHNEN
440 GOSUB 60000
450 NEXT I
999 END
```

```
1000 POKE 12512,1:POKE 12513,62:POKE 12514,100:POKE 12515,164
1010 POKE 12516,36:POKE 12517,36:POKE 12518,36:POKE 12519,36
1020 RETURN
2000 IF X<0THEN X=0
2010 IF X>319 THEN X=319
2020 IF Y<0THEN Y=0
2030 IF Y>199 THEN Y=199
2040 RETURN
60000 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
60020 XK=8*INT(X/8)
60040 EX=2↑(7-INT((X/8-INT(X/8))*8))
60060 S=8192+YK+XK
60100 POKES,PEEK(S) OR EX
60120 RETURN

READY.
```

Y1=SIN(X)*COS(4*X)

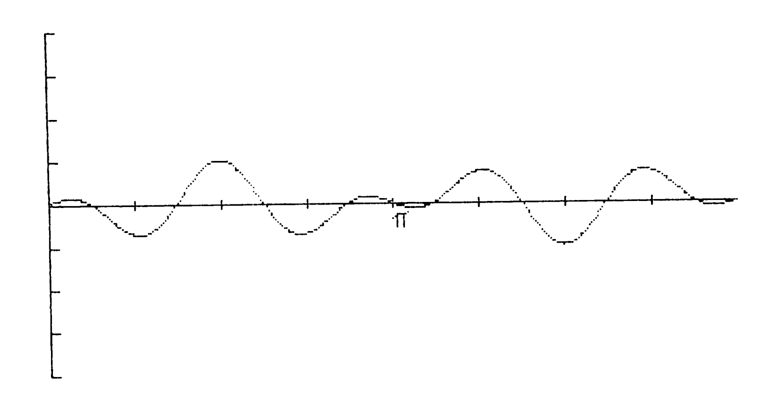


Abb. 5.8

5.9. Drei-dimensionale Grafiken in hochauflösender Grafik

Zum Schluß nun das Programm, mit dem z.B die folgenden Abbildungen erstellt wurden. Dafür brauchen Sie allerdings viel Geduld! Die "schnellste" Abbildung benötigt immerhin ca. eine halbe Stunde bis zur Vollendung. (Die Zeit ist natürlich auch von der Schrittweite abhängig, mit der die Grafik gezeichnet wird).

Wie im vorigen Abschnitt bleibt das Hauptprogramm für die verschiedenen Motive gleich. Geändert werden müssen nur die Funktionsgleichungen (Zeile 160 - 170) und die Anfangswerte (Zeile 140). Die Schrittweite DX (Zeile 210) beeinflußt die Dichte der Darstellung. (Mit der im Programm angegebenen Schrittweite läßt sich das Motiv sehr gut erkennen, mit einer größeren Schrittweite hingegen können Sie die Zeichen-Zeit entsprechend verkürzen).

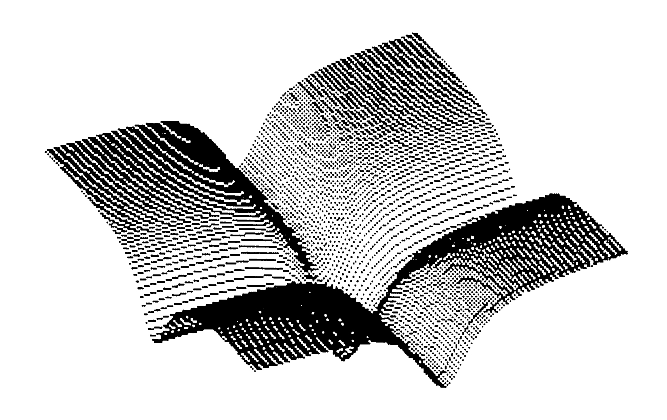


Abb. 5.9

```
10 REM.................................DREIDIMENSIONALE BEISPIELE
20 REM.................................GRAPHIK EINSCHALTEN
50 V=53248
60 POKE V+17,59: POKE V+24,24
70 FORI=1024TO2023: POKE I,14: NEXTI
80 FOR I=8192TO16383: POKE I,0: NEXTI
90 REM.................................FUNKTIONSGLEICHUNG
140 X1=-4: X2=8: Y1=-4: Y2=8: VA=2: A=2
160 DEF FNZ(X)=1/(SIN(X)+1.08)+1/(SIN(Y)+1.08)
170 DEF FNT(X)=0
180 NY=A*87: NX=A*50
190 DX=(X2-X1)/NX: DY=(Y2-Y1)/NY
200 H0=0
210 FOR X=X1TOX2 STEPDX+.3
220 H0=(X-X1)/(X2-X1)*100: NH=-1
230 FOR Y=Y1TOY2 STEPDY
240 ZZ=FNZ(X)+FNT(X)
250 YY=-ZZ*VA-(Y-Y1)/(Y2-Y1)*58+90+H0
260 NH=NH+1
270 XX=H0+NH*174/NY
280 IF YY<0 OR YY>199 THEN GOTO 300
290 GOSUB 60000
300 NEXT Y
310 NEXT X
320 INPUTA$
340 POKE V+17,155: POKE V+24,21: PRINT "♥": END
60000 XK=8*INT(XX/8)
60020 YK=320*INT(YY/8)+INT((YY/8-INT(YY/8))*8)
60040 EX=2↑(7-INT((XX/8-INT(XX/8))*8))
60060 S=8192+XK+YK
60080 POKE S,PEEK(S) OR EX
60100 RETURN

READY.
```

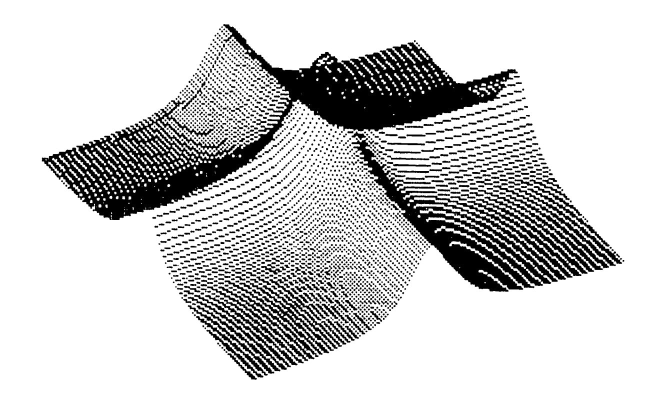


Abb. 5.10

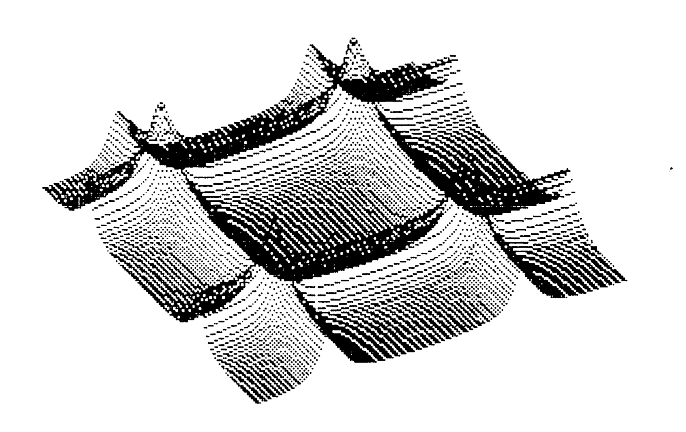


Abb. 5.11

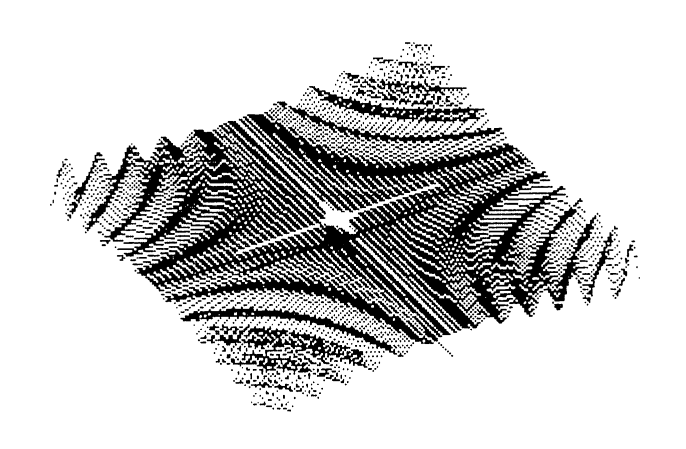


Abb. 5.12

Die Abbildungen 5.9 bis 5.12 wurden mit folgenden Definitionen erzeugt:

Abb. 5.9:

```
DEF FN Z(X) = 10 -1/(SIN(X)+1.08) -1/(SIN(Y)+1.08)
DEF FN T(X) = 0
X1 = -4: X2 = .5: Y1 = -4: Y2 = .5: VA = 3: A = 2
```

Abb. 5.10:

```
DEF FN Z(X) = 1/(SIN(X)+1.08) + 1/(SIN(Y)+1.08)
DEF FN T(X) = ((ABS(SIN(X))< .995) OR
              (ABS(SIN(Y))< .995)) * 2
X1 = -4: X2 = 1: Y1 = -4: Y2 = 1: VA = 3: A = 2
```

Abb. 5.11:

```
DEF FN Z(X) = 1/(SIN(X)+1.08) + 1/(SIN(Y)+1.08)
DEF FN T(X) = 0
X1 = -4: X2 = 8: Y1 = -4: Y2 = 8: VA = 2: A = 2
```

Abb. 5.12:

```
DEF FN Z(X) = SIN(X*Y) * X * Y/36
DEF FN T(X) = (ABS(X*Y)> ABS(DX)/2)
X1 = - 6: X2 = 7: Y1 = -6: Y2 = 6: VA = 10: A = 2
```

6. BASIC - Schlüsselwörter

Diese Übersicht ist nicht vollständig, sondern enthält nur die im vorliegenden Buch verwendeten BASIC-Anweisungen des Commodore 64.

ABS(X)	Ergibt den Absolutwert des Argumentes X.
AND	Logisches 'und', der durch AND verknüpfte Ausdruck ist nur dann wahr, wenn alle Teilausdrücke wahr sind.
ASC(A$)	Ergibt den ASCII-Code des ersten String-Zeichens.
CHR$(X)	Umkehrfunktion zu ASC. Ergibt das dem betreffenden ASCII-CODE zugeordnete Zeichen.
DATA	Ermöglicht das Speichern von Daten in BASIC-Programmen. Die Daten werden mit der READ-Anweisung gelesen.
DEF FN	Damit kann eine eigene Funktion mit einer Variablen definiert werden.
END	Bewirkt das Beenden eines Programmlaufs.
EXP(X)	Entspricht e↑x, wobei e = 2.71827183.
FOR .. TO .. STEP ..	Erlaubt das wiederholte Durchlaufen eines Programm-Teils. Anzugeben ist der Anfangs- und der Endwert der Schleifenvariablen. Die Schrittweite beträgt 1, wenn sie nicht (mit STEP) ausdrücklich auf einen anderen Wert gesetzt wird.
GET	Liest ein Zeichen von der Tastatur (ohne Eingabe von RETURN).
GOSUB	Bewirkt einen Sprung zu dem angegebenen Unterprogramm. Anschließend wird das Programm mit der auf GOSUB folgenden Anweisung fortgesetzt.
GOTO	Unbedingter Sprung zu der angegebenen Zeile. Diese Anweisung sollte sparsam verwendet werden, da sie zu unübersichtlichen Programmen

führt. Leider läßt sie sich in BASIC oft kaum vermeiden.

IF .. THEN ..	Nur dann, wenn die nach IF stehende Bedingung wahr ist, werden die nach THEN stehenden Anweisungen ausgeführt.
INPUT	Weist einen über die Tastatur eingegebenen Wert der nach INPUT stehenden Variablen zu, erlaubt also, Daten im Dialog in ein Programm einzugeben.
NEXT	Schließt eine mit FOR eröffnete Schleife ab.
PEEK	Liest den Wert aus der angegebenen Speicheradresse.
POKE S,N	Schreibt den Wert N in die Speicherzelle S. Da der Commodore 64 standardmäßig über keine spezifischen Grafik-Befehle in BASIC verfügt, leider der meistgebrauchte Befehl in diesem Buch!
PRINT	Ausgabebefehl.
READ	Liest der Reihe nach die in DATA-Zeilen stehenden Daten ein.
RESTORE	Ermöglicht das wiederholte Lesen von DATA-Zeilen.
RETURN	Bewirkt den Rücksprung von einem Unterprogramm zu der Stelle, von der es aufgerufen wurde.
SIN	Sinusfunktion.
SPC(X)	Erzeugt X Leerzeichen.
SQR(X)	Berechnet die Quadratwurzel von X.
TAB	Tabulatorfunktion. In Verbindung mit PRINT kann die Ausgabespalte gewählt werden.

7. Tabellen

7.1. Speicherbelegungs-Tabelle

Wie schon erwähnt, hat der der Commodore 64 mehr als 64000 Speicherplätze (genau: 64 K, d.h. 64 x 1024, also 65536). Diese Speicherplätze lassen sich aufgrund der Aufgaben, die sie im Computer erfüllen, zu verschiedenen Gruppen zusammenfassen.

Die folgende Tabelle soll Ihnen eine Übersicht über die Speicherbelegung Ihres Computers geben. Die Speicherplatzliste im Commodore-Handbuch (ab S. 160) gibt leider nur die Funktionen der Speicher bis Nr. 53247 an, für die Graphik sind aber gerade die Speicher ab Nr. 53248 von Bedeutung.

Speicherplätze 0 - 827: Diese Speicher sind notwendig für den Betrieb des Computers (System-Adressen).

828 - 1019: Hier befindet sich der Kassettenpuffer. Alle Daten, die von der Kassette übernommen werden, werden zunächst hier zwischengespeichert, bevor sie z.B. im Programm weiterverarbeitet werden. Diese Speichergruppe wird also nur bei Kassetten-Betrieb benötigt. Arbeitet man hingegen mit einem Disketten-Laufwerk, können hier Daten oder kurze Maschinen-Programme abgelegt werden.

1024 - 2023: Diese Speicherplätze belegt der Bildschirmspeicher (Tabelle 7.2).

Daran schließt sich die Speichergruppe 2040 - 2047 an, die bei der Verwendung von Sprites gebraucht werden.

Von 2048 - 40959 (= 38 K) folgt der RAM-Bereich des Computers (RAM = 'random access memory': Speicher mit freier Zugriffsmöglichkeit). Hier können Daten und Programme gespeichert werden. Dieser Speicherbereich ist allerdings "flüchtig", d.h. beim Abschalten des Computers wird der ganze RAM-Bereich gelöscht.

Darin liegt der wichtigste Unterschied zum ROM-Speicher (von 40960 - 49151). Dieser Bereich bleibt permanent erhalten, und ist auch vom Benutzer nicht zu

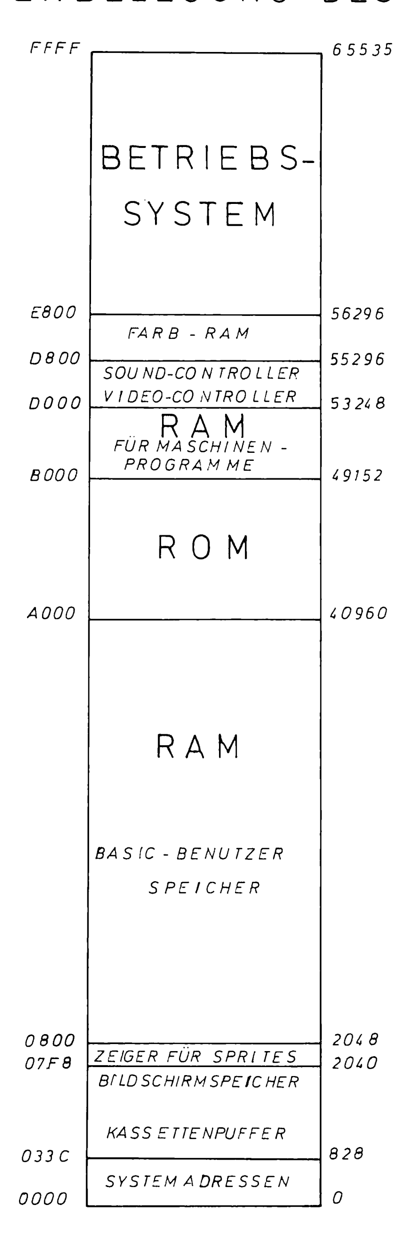
verändern. (ROM = 'read only memory': aus diesem Bereich kann nur gelesen werden). Hier liegt z.B. der Interpreter, der es dem Computer ermöglicht, BASIC-Programme zu "verstehen" und auszuführen.

Der folgende RAM-Bereich steht zwar dem Benutzer zur Verfügung, kann allerdings nicht für BASIC- sondern nur für Maschinen-Programme verwendet werden.

Die Speichergruppe von 53248 - 55295 ist zuständig für die Aufgaben des Video- und des Sound-Chips.

Darauf folgt der Bereich des Farbspeichers. Die letzten Speicherplätze (56296 - 65535) können wieder nicht gelöscht werden. Sie sind zuständig für die interne Organisation des Computers und stellen die Maschinenspracheroutinen zum Betrieb des Rechners zur Verfügung.

SPEICHERBELEGUNG DES COMMODORE 64



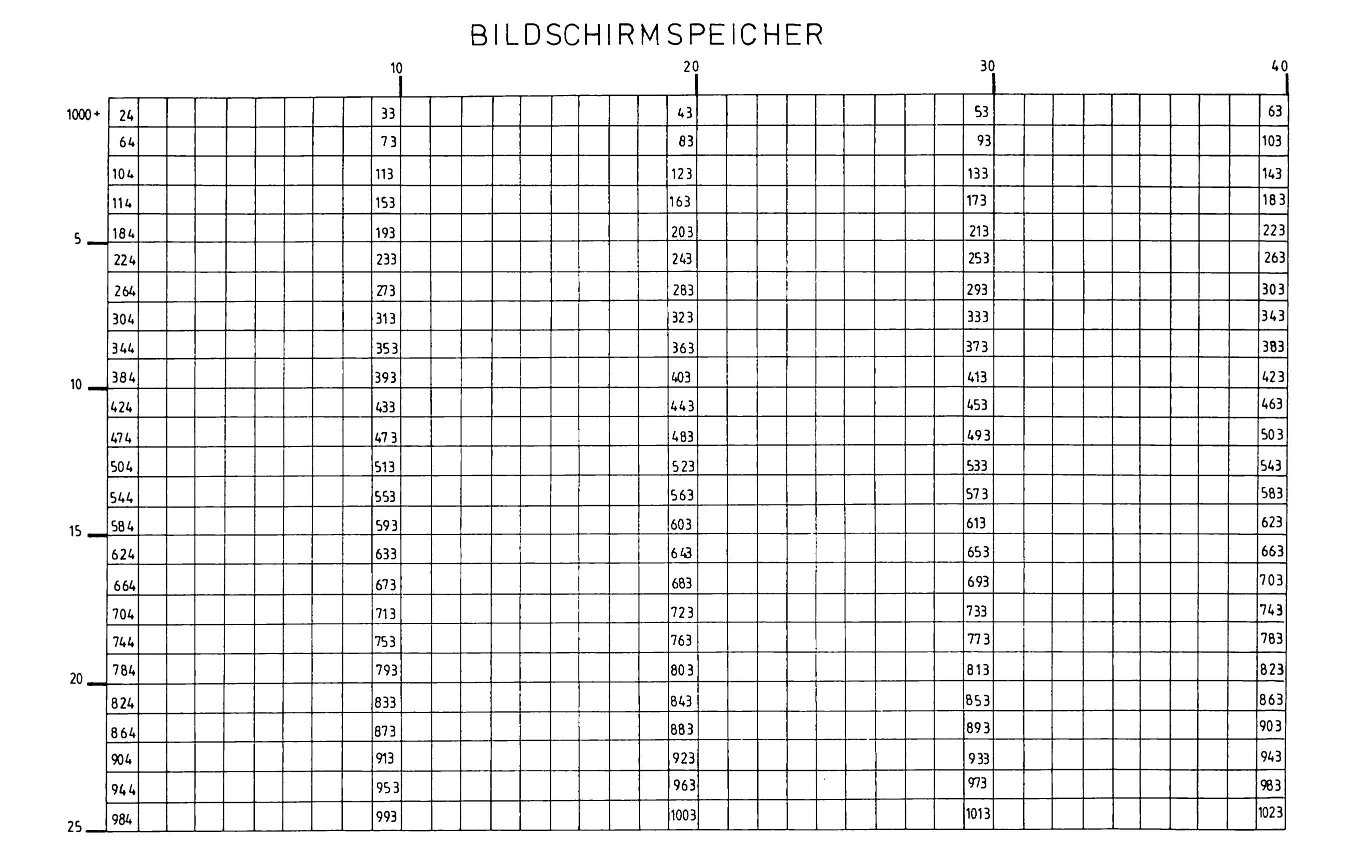
7.2. Der Bildschirmspeicher

Die Tabelle zeigt die Verteilung der Speicheradressen von 1024 bis 2023, also der Speichergruppe, die für die Darstellung von Zeichen an den entsprechenden Bildschirmplätzen sorgt. Um eine bestimmte Bildschirmstelle zu finden, muß man zu den angegebenen Werten 1000 addieren und bis zur gewünschten Stelle weiterzählen.

Beispiel:

```
10 FOR I = 1 TO 40
20 POKE 1024+I,81
30 NEXT I
```

Durch diese Anweisungen wird in der obersten Bildschirmzeile eine Reihe von Kreisen erzeugt, die allerdings bisher noch unsichtbar sind, da sie zunächst in der Hintergrundfarbe dargestellt werden. Um sie sichtbar zu machen, muß noch eine andere Farbe (als die Hintergrundfarbe) in die entsprechenden Farbspeicher gebracht werden (s. Farbspeicher-Tabelle).



BILDSCHIRMSPEICHER
10
20
30
40
1000+
5
10
15
20
25
24 33 43 53 63
64 73 83 93 103
104 113 123 133 143
114 153 163 173 183
184 193 203 213 223
224 233 243 253 263
264 273 283 293 303
304 313 323 333 343
344 353 363 373 383
384 393 403 413 423
424 433 443 453 463
474 473 483 493 503
504 513 523 533 543
544 553 563 573 583
584 593 603 613 623
624 633 643 653 663
664 673 683 693 703
704 713 723 733 743
744 753 763 773 783
784 793 803 813 823
824 833 843 853 863
864 873 883 893 903
904 913 923 933 943
944 953 963 973 983
984 993 1003 1013 1023

7.3. Der Farbspeicher

Der Farbspeicher befindet sich in den Adressen von 55296 bis 56295 (= 1000 Speicherplätze, d.h. ein Speicherplatz für jede der 40 x 25 Bildschirmpositionen). Aus diesen Speicherplätzen entnimmt der Computer die Information, welche Farbe ein (beliebiges) Zeichen an einer bestimmten Stelle des Bildschirms haben soll.

Ein Zeichen kann auf dem Bildschirm nur dann erscheinen, wenn außer der Bildschirmposition auch eine Farbe für die betreffende Stelle angegeben wird. Ohne eine Farbangabe werden die Zeichen in der Hintergrundfarbe dargestellt - sind also nicht zu sehen!

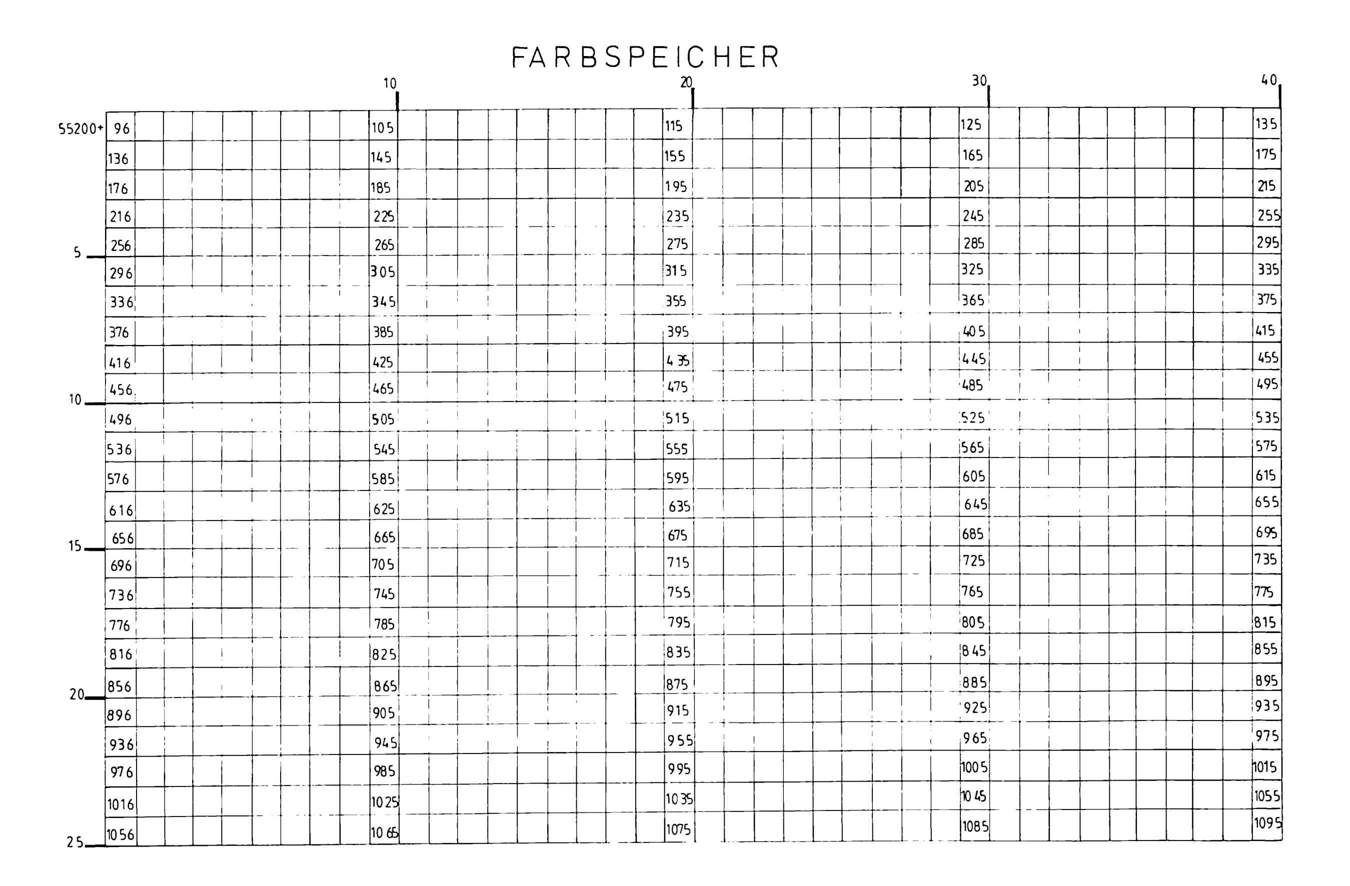
Das Beispiel in der Beschreibung der Bildschirmspeicher-Tabelle muß also um eine Farbangabe erweitert werden, z.B. durch:

```
25 POKE 55296+I,7
```

Jetzt wird die Reihe aus Kreisen in gelber Farbe sichtbar.

Die Code-Werte für die Farben finden Sie in der Übersicht hinter der Farbspeicher-Tabelle.

FARBSPEICHER



FARBCODES

0	SCHWARZ
1	WEIß
2	ROT
3	TÜRKIS
4	VIOLETT
5	GRÜN
6	BLAU
7	GELB
8	ORANGE
9	BRAUN
10	HELLROT
11	GRAU 1
12	GRAU 2
13	HELLGRÜN
14	HELLBLAU
15	GRAU 3

7.4. Der Grafik-Speicher

Für die Darstellung in der hochauflösenden Grafik ist der Grafik-Speicher (8192 bis 16191) zuständig. Durch die Werte dieser Speicherplätze wird bestimmt, an welchen Bildschirmstellen Punkte gesetzt werden.

Die erste Speicheradresse (8192) bezieht sich auf die erste Reihe des normalen Bildschirmkästchens in der linken oberen Ecke, die achte Adresse bezieht sich auf die letzte Reihe dieses Kästchens, die neunte auf die erste Reihe des rechts daneben liegenden, usw. Diese Art der Speicheraufteilung weicht deutlich von der von den Sprites bekannten ab!

Um Ihnen das Auffinden der Speicherplätze zu erleichtern, z.B. wenn Sie einzelne Zeichen definieren wollen, ist die Speichereinteilung in der folgenden Tabelle dargestellt.

Die Aufgabe eines Farbspeichers übernimmt in der hochauflösenden Darstellung der Bildschirmspeicher. (Deshalb erscheinen die Systemmeldungen im Grafik-Modus in Farbe.)

GRAFIKSPEICHER

10 20 30 40

8192	192	200	240	280	320	360	400	440	480	504
8512	512	520	560	600	640	680	720	760	800	824
8832	832	840	880	920	960	1000	1040	1080	1120	1144
9152	152	160	200	240	280	320	360	400	440	464
9472	472	480	520	560	600	640	680	720	760	884
9792	792	800	840	880	920	960	1000	1040	1080	1104
10112	112	120	160	200	240	280	320	360	400	424
10432	432	440	480	520	560	600	640	680	720	744
10752	752	760	800	840	880	920	960	1000	1040	1064
11072	072	080	120	160	200	240	280	320	360	384
11392	392	400	440	480	520	560	600	640	680	704
11712	712	720	760	800	840	880	920	960	1000	1024
12032	032	040	080	120	160	200	240	280	320	344
12352	352	360	400	440	480	520	560	600	640	664
12672	672	680	720	760	800	840	880	920	960	984
12992	992	1000	1040	1080	1120	1160	1200	1240	1280	1304
13312	312	320	360	400	440	480	520	560	600	624
13632	632	640	680	720	760	800	840	880	920	944
13952	952	960	1000	1040	1080	1120	1160	1200	1240	1264
14272	272	280	320	360	400	440	480	520	560	584
14592	592	600	640	680	720	760	800	840	880	904
14912	912	920	960	1000	1040	1080	1120	1160	1200	1224
15232	232	240	280	320	360	400	440	480	520	544
15552	552	560	600	640	680	720	760	800	840	864
15872	872	880	920	960	1000	1040	1080	1120	1160	1184

7.5. Das Sprite-Entwurfsblatt

Das Sprite-Entwurfsblatt besteht aus einem Raster von 24x21 Kästchen (entsprechend der Größe eines Sprites). Das Raster ist nochmal in drei Gruppen unterteilt. Jedes Raster-Kästchen entspricht einem Sprite-Punkt. Dazu gehört eine Tabelle für die ausgerechneten Daten.

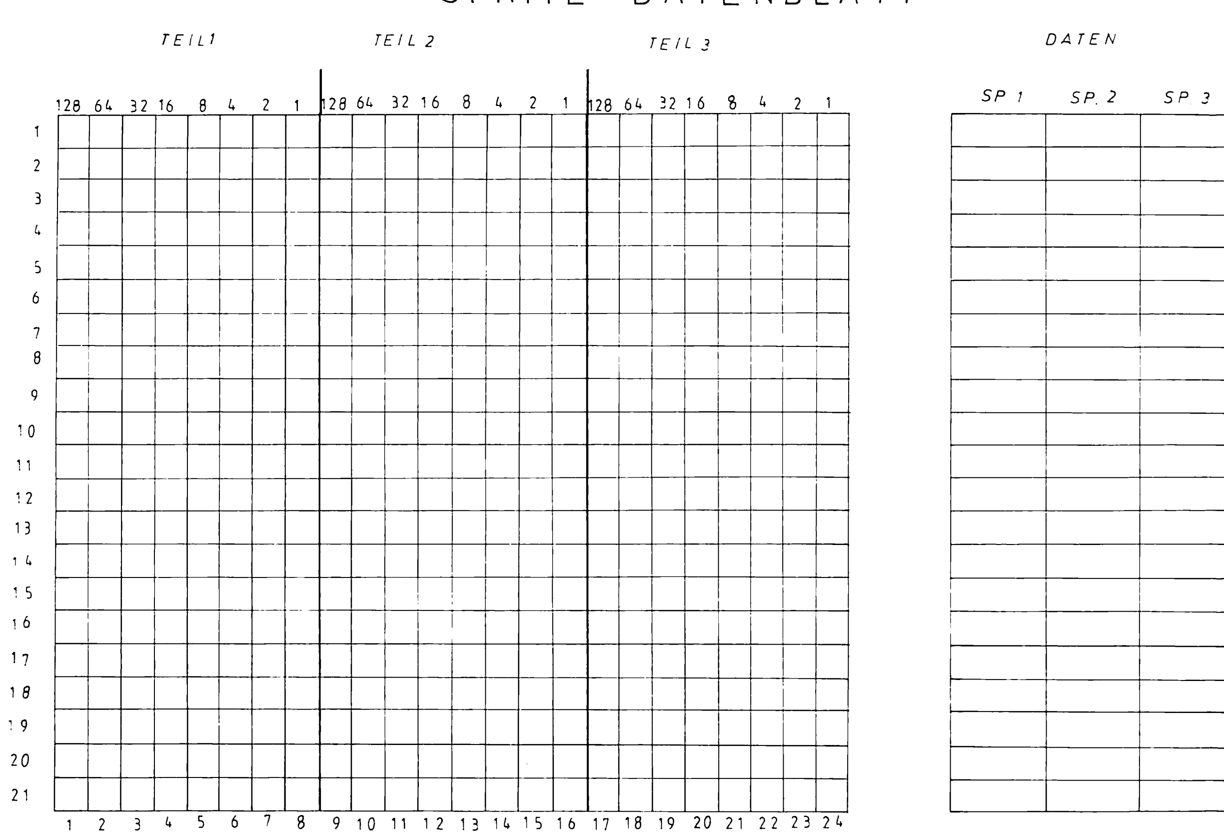
Hat man nun eine Figur in das Raster eingezeichnet und alle betroffenen Kästchen ausgefüllt, müssen die Daten für den Computer ausgerechnet werden. Dabei geht man zeilenweise von oben nach unten und innerhalb der Zeilen von links nach rechts vor:

Die Zahlen am oberen Rand der Tabelle, die über ausgefüllten Kästchen einer Gruppe stehen, werden addiert und in die Datentabelle übertragen.

Ist die Datentabelle vollständig ausgefüllt, kann man das Sprite - wie in Kapitel 2. beschrieben - auf dem Bildschirm erscheinen lassen.

In Kapitel 5 befindet sich ein Programm, mit dem sich Sprites auf dem Bildschirm entwerfen lassen. Dabei übernimmt der Computer das Berechnen der Daten.

SPRITE - DATENBLATT



7.6. Die Commodore-Zeichen

Die folgenden Seiten geben eine Übersicht über die Commodore-Grafik-Zeichen.

In der Tabelle ist angegeben, auf welcher Buchstaben-Taste das jeweilige Zeichen zu finden ist, sowie der CHR$-Wert des Zeichens und die POKE-Werte des Zeichens in der normalen und inversen Darstellung. Hierbei wurden einige im Commodore-Handbuch vorkommende Abweichungen zwischen CHR$-Wert und dazugehörigem Zeichen richtiggestellt.

POKE und CHR $ - WERTE

ZEICHEN	TASTE	POKEWERT normal	invers	CHR $
		64	192	96
	A	65	193	97
	B	66	194	98
	C	67	195	99
	D	68	196	100
	E	69	197	101
	F	70	198	102
	G	71	199	103
	H	72	200	104
	I	73	201	105
	J	74	202	106
	K	75	203	107
	L	76	204	108
	M	77	205	109
	N	78	206	110
	O	79	207	111
	P	80	208	112
	Q	81	209	113
	R	82	210	114
	S	83	211	115
	T	84	212	116

ZEICHEN	TASTE	POKEWERT normal	invers	CHR $
	U	85	213	117
	V	86	214	118
	W	87	215	119
	X	88	216	120
	Y	89	217	121
	Z	90	218	122
		91	219	123
		92	220	124
		93	221	125
		94	222	126
		95	223	127
LEERZEICHEN		96	224	
		97	225	161
		98	226	162
		99	227	163
		100	228	164
		101	229	165
		102	230	166
		103	231	167
		104	232	168
		105	233	169

ZEICHEN	TASTE	POKEWERT normal	inverts	CHR $
		106	234	170
		107	235	171
		108	236	172
		109	237	173
		110	238	174
		111	239	175
		112	240	176
		113	241	177
		114	242	178
		115	243	179
		116	244	180
		117	245	181
		118	246	182
		119	247	183
		120	248	184
		121	249	185
	☑	122	250	186
		123	251	187
		124	252	188
		125	253	189
		126	254	190
		127	255	191

7.7. Die Speichertabellen für Sprites

Auf den folgenden Seiten sind die Speicherstellen des Video-Chips zusammengestellt, die sich auf die Programmierung von Sprites beziehen. Einzelheiten über die Funktionsweise dieser Speicher sind in Kapitel 2 beschrieben.

SPEICHERTABELLEN - SPRITES (V=53248)

V+			
0 1	SPRITE	0	X -KOORDINATE Y -KOORDINATE
2 3	SPRITE	1	X - '' Y - ''
4 5	SPRITE	2	X - '' Y -
6 7	SPRITE	3	X - '' Y -
8 9	SPRITE	4	X - '' Y -
10 11	SPRITE	5	X - '' Y -
12 13	SPRITE	6	X - '' Y -
14 15	SPRITE	7	X - '' Y -

V+			
39	SPRITE	0	FARBE
40	SPRITE	1	''
41	SPRITE	2	''
42	SPRITE	3	''
43	SPRITE	4	''
44	SPRITE	5	''
45	SPRITE	6	''
46	SPRITE	7	''

V+16

	SPRITE: X-KOORDINATE über 255							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+21								
	SPRITE: EIN-UND AUSSCHALTEN							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+23								
	SPRITE: IN Y-VERGRÖSSERN							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+27								
	SPRITE: HINTERGRUND-PRIORITÄT							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+28								
	SPRITE: MEHRFARBIG							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+29								
	SPRITE: IN X-VERGRÖSSERN							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+30								
	SPRITE: MIT SPRITE KOLLISION							
	128	64	32	16	8	4	2	1
	7	6	5	4	3	2	1	0

V+31

	SPRITE:HINTERGRUND KOLLISION							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0
V+37								
	SPRITE:MEHRFARBEN NR.: 0							
	FARBCODES							
	FÜR ALLE SPRITES							
V+38								
	SPRITE:MEHRFARBEN NR.: 1							
	FARBCODES							
	FÜRALLE SPRITES							

	2047	2046	2045	2044	2043	2042	2041	2040
NR.:	7	6	5	4	3	2	1	0

S t i c h w o r t v e r z e i c h n i s

Herrmanns CBM Programmsammlung in Basic

Band 1: Spiele, Knobeleien und Simulationen

Von Dietmar Herrmann, Anzing, 232 Seiten, 1982

ISBN 3-88322-013-2, kart., DM 32.–

Der erste Band einer neuen Reihe, die mit zahlreichen Programmen für Spiele und "ernsthafte" Themen den Commodore-Computer dem Benutzer näherbringt. Dietmar Herrmann hat aus seiner Schulpraxis heraus Programme entwickelt, die das Lernen und Spielen mit dem Computer zum Vergnügen machen.





Herrmanns CBM Programmsammlung in Basic

Band 2: Wirtschaft

Von Dietmar Herrmann, Anzing, ca. 180 Seiten, 1983

ISBN 3-88322-014-0, kart., DM 32.–

Jetzt wird es ernst: Hier wird Ihnen gezeigt, wie Sie Ihren Commodore-Computer für sich arbeiten lassen können: Er hilft Ihnen z.B., Ihren Lohnsteuer-Jahresausgleich oder die Einkommensteuer-Erklärung zu erledigen, zeigt Ihnen grafisch, wohin die Staatsverschuldung geht — oder auch Ihre eigene, berechnet Ihre Zinsen (Soll oder Haben) auf der Bank, oder wie Sie Ihr Haus finanzieren können.

Als nächstes folgt: **Band 3: Mathematik**

TRS-80 Assembler-Programmierung

Von Dipl. Ing. Günther Daubach, Leverkusen, ca. 180 Seiten,
In Vorbereitung 1983,

ISBN 3-88322-017-5, kart., ca. DM 48.–

Für den Anfänger eine verständliche Einführung in die 'Muttersprache' des TRS-80 mit zahlreichen Erklärungen der Befehle an den Z80 Mikroprozessor, für den Kenner der Assemblerprogrammierung eine Erweiterung seines Wissens über Details, z.B. die Verwendung nützlicher ROM-Routinen und die Speicherung von Variablen, Datenaufzeichnung auf Kassette und Diskette, sowie auch die Unterschiede zwischen TRS-80, Modell I und III.

Programmkassetten und -disketten
zu diesem Buch

GRAPHIK AUF DEM COMMODORE 64

Warum sich Arbeit machen, die andere schon für Sie erledigt haben ?

Die Programme - zum Teil in erweiterter Form - aus diesem Buch können Sie auf Disketten oder Kassetten zusammen mit einer ergänzenden Beschreibung beziehen.

Grafik-Kassette: Best.-Nr. 880 22 5 01 DM 58.-
Grafik-Diskette: Best.-Nr. 880 22 1 01 DM 58.-

Die Preise enthalten die ges. MwST.

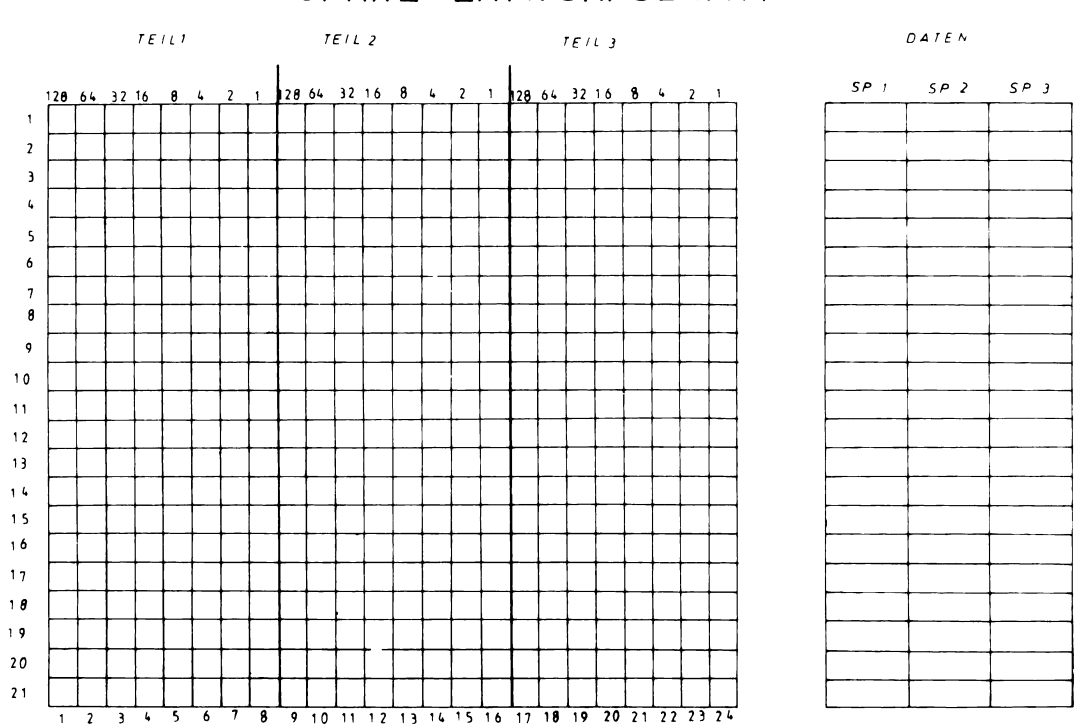
Ihre Bestellung richten Sie bitte an:

IWT Software Service GmbH

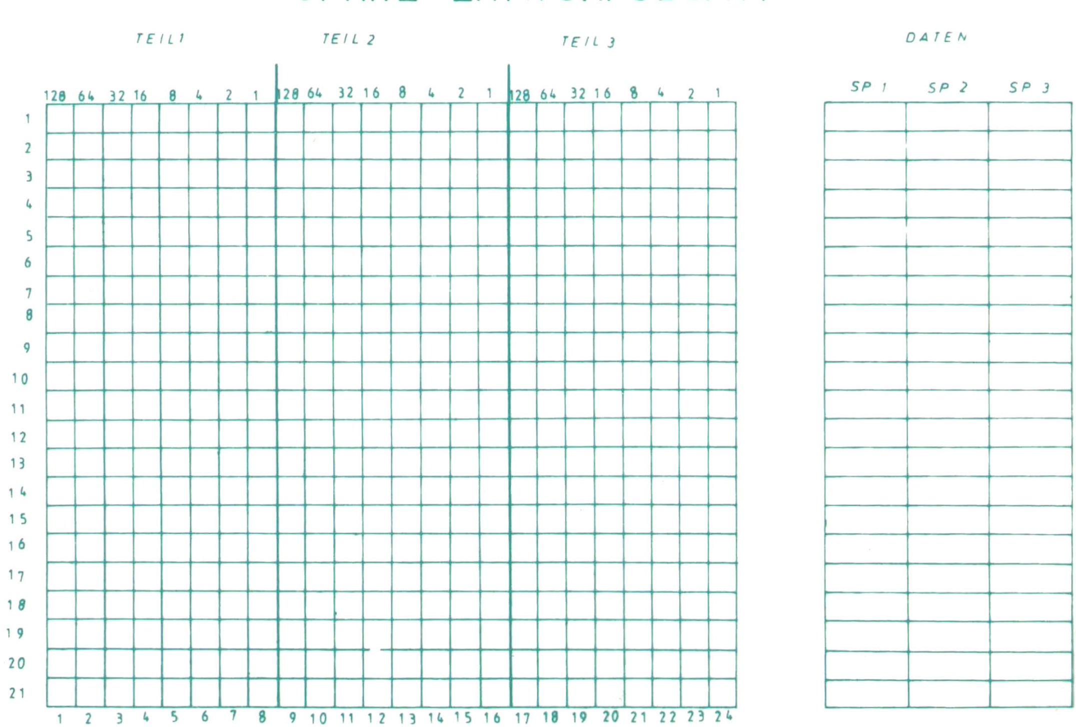
Altenberger Str. 23 b
5093 Burscheid
Tel.: 02174/62815
Telex: 5213989 iwt d

Verkaufsbüro:
Dahlienstraße 4
8011 Vaterstetten
Tel.: 08106/4986
Telex: 5213989 iwt d

SPRITE-ENTWURFSBLATT



SPRITE - ENTWURFSBLATT



J. Elsing, H. Sterner, A. Wagner

Grafik auf dem Commodore 64

Der Commodore 64 bietet für einen Computer seiner Preisklasse vielseitige grafische Möglichkeiten.
Dieses Buch gibt dem Leser Informationen, wie er die Grafikfunktionen anwenden kann – Informationen, die er im Commodore-Handbuch nicht findet.
Ausgehend von einfachen Grafiken mit den „fest eingebauten" Grafik-Zeichen des Commodore 64, führt das Buch systematisch zu den anspruchsvolleren grafischen Gestaltungsmöglichkeiten dieses Computers, illustriert jeweils durch ein typisches Beispiel-Programm. Unter anderem findet man ein Hilfsprogramm zur Erstellung der „Sprites"-Grafik, einer besonderen Grafik-Möglichkeit dieses Computers zur Erzeugung beweglicher Figuren.
Zahlreiche Tabellen und Übersichten runden die Darstellungen ab. Die beiliegende Folie hilft bei der Erstellung von Sprite-Figuren.

ISBN 3-88322-027-2